HiGEC: Hierarchy Generation and Exploitation for Enhanced Multi-Class Classification

CELAL ALAGÖZ, Sivas University of Science and Technology, Türkiye

Background: Hierarchical Classification (HC) has long been recognized for improving predictive performance by exploiting relationships between classes. However, most tabular multi-class datasets lack predefined class hierarchies, limiting the broader applicability of hierarchy-aware learning methods.
Objectives: This study introduces HiGEC (Hierarchy Generation and Exploitation for Classification), a unified framework that enables hierarchy-aware learning for standard flat-label tabular classification problems through automatically generated class hierarchies. The objectives are to determine whether data-driven hierarchies can consistently improve performance over Flat Classification (FC), whether hierarchy-aware evaluation metrics improve under enhanced exploitation strategies, and whether probabilistic aggregation mitigates hierarchical error propagation.
Methods: HiGEC systematically integrates Hierarchy Generation (HG) and Hierarchy Exploitation (HE) within a unified framework. Two enhanced HE schemes are proposed: HE+, which mitigates error propagation through probabilistic path aggregation, and HE+F, which combines hierarchical and flat predictions through calibrated convex fusion. A large-scale benchmark involving 100 tabular datasets and ten classifiers—including gradient boosting, ensemble, generative, instance-based, and transformer-based methods—was conducted. Evaluation included both conventional classification metrics and hierarchy-aware metrics, together with statistical significance testing and runtime analysis.
Results: HiGEC consistently improved predictive performance over FC baselines, particularly in F1-score, with statistically significant gains across diverse classifiers. Hierarchy-aware evaluation further demonstrated improvements in structural consistency, with enhanced HE schemes achieving higher hierarchical F-measure (hF) and lower path-based loss than baseline hierarchical methods. Runtime analysis showed that optimized HE+F configurations typically incurred moderate computational overhead (approximately 2–3× slower than FC), while certain lightweight HE+ configurations achieved both higher predictive performance and lower runtime than their FC counterparts.
Conclusions: The empirical findings support the three formal hypotheses of this study: (1) HiGEC improves flat predictive performance over FC, (2) enhanced HE strategies improve hierarchy-aware evaluation metrics, and (3) probabilistic aggregation mitigates hierarchical error propagation. Factor-wise ranking analysis further demonstrated that HE strategies contribute more strongly to performance variation than HG methods under aggregated evaluation. Overall, HiGEC provides a unified and reproducible framework for hierarchy-aware learning on tabular multi-class benchmarks.

JAIR Associate Editor: Julia Handl

JAIR Reference Format:
Celal Alagöz. 2026. HiGEC: Hierarchy Generation and Exploitation for Enhanced Multi-Class Classification. *Journal of Artificial Intelligence Research* 86, Article 42 (August 2026), 36 pages. DOI: 10.1613/jair.1.22925

1 Introduction

Classification is a fundamental task in Machine Learning (ML) that aims to assign labels to previously unseen instances using labeled training data. Although many learning algorithms naturally extend to multi-class settings, multi-class classification remains challenging in problems involving large label spaces, overlapping classes, or complex inter-class relationships.

Author's Contact Information: Celal Alagöz, ORCID: 0000-0001-9812-1473, celal.alagoz@gmail.com, Sivas University of Science and Technology, Sivas, Türkiye.

DOI: 10.1613/jair.1.22925

A widely adopted strategy for addressing multi-class complexity is decomposition into simpler sub-problems. Traditional approaches such as one-versus-all and all-versus-all decompose the task into multiple binary classifiers (Aly 2005). An alternative paradigm is HC, where classes are organized into a tree-like structure that recursively partitions the label space into smaller and semantically related subsets (Godbole 2002; Vural and Dy 2004). Such hierarchical decomposition can improve scalability, interpretability, and predictive performance, particularly in domains with naturally defined taxonomies, including text categorization (Onan 2023; A. Sun and Lim 2001; Zeng et al. 2025), biomedical applications (Hsu and Tseng 2024; Peng et al. 2024), and image analysis (Liu and Wang 2024; Shi et al. 2023). In contrast, this study focuses specifically on tabular multi-class classification, where such semantic structures are typically unavailable. Tabular data—characterized by heterogeneous features and the absence of spatial or sequential relationships—lack inherent organization, making automated hierarchy generation essential. Consequently, this study investigates whether data-driven hierarchies can improve both predictive performance and structural consistency on general flat-label tabular benchmarks.

Despite these advantages, most real-world multi-class datasets do not include predefined hierarchies. Constructing meaningful hierarchies manually is often infeasible, motivating the development of automated HG techniques. Prior studies have shown that automatically generated hierarchies can occasionally improve classification performance (Babbar et al. 2016; T. Li et al. 2007); however, empirical findings remain inconsistent (Del Moral, Nowaczyk, and Pashami 2022; Del Moral, Nowaczyk, Sant'Anna, et al. 2023; Helm, W. Yang, et al. 2021). Existing studies typically evaluate limited combinations of hierarchy generation and exploitation strategies, rely on small benchmarks, and assess improvements primarily through isolated flat performance metrics. Consequently, it remains unclear whether hierarchy-aware learning consistently provides measurable benefits over FC, and which components of the hierarchical pipeline are responsible for observed improvements.

To address these limitations, this study introduces HiGEC, a unified framework that systematically integrates automatic HG with extended HE strategies for standard flat multi-class problems. The framework consists of two sequential phases, illustrated in Figure 1. In the HG phase, class hierarchies are automatically constructed using data-driven dissimilarity measures derived either from representative statistics or classifier behavior. In the HE phase, the generated hierarchy is exploited using enhanced local and global classification schemes designed to mitigate hierarchical error propagation and improve prediction consistency through probabilistic aggregation and flat–hierarchical fusion mechanisms.

In this work, the term *HE* specifically refers to the utilization of a hierarchy during prediction and decision-making. While the HC literature often assumes predefined semantic hierarchies, HiGEC explicitly separates hierarchy construction from hierarchy utilization. This distinction enables investigation of automatically generated hierarchies as instrumental structures for improving predictive performance, regardless of whether semantic taxonomies are available.

This motivates three formal hypotheses that guide the empirical investigation:

- **H1 (Performance Gain):** Automatically generated hierarchies, when combined with optimized HE strategies, achieve statistically significant improvements over FC baselines in macro F1-score across tabular multi-class datasets.
- **H2 (Hierarchical Quality):** Enhanced HE strategies (HE+ and HE+F) improve hierarchy-aware evaluation metrics, including hierarchical F-measure (hF) and path-based loss, relative to baseline hierarchical methods.
- **H3 (Error Mitigation):** Probabilistic path aggregation and flat–hierarchical fusion mitigate hierarchical error propagation by reducing structurally inconsistent predictions and path-based hierarchical error.

To reduce the risk of post-hoc interpretation, hyperparameter exploration was conducted on a held-out development subset prior to the final benchmark evaluation.

To evaluate these hypotheses, the proposed framework is assessed using one of the largest HC-oriented benchmarks to date, comprising 100 OpenML multi-class datasets and ten heterogeneous classifiers. Unlike

previous studies, the evaluation includes both flat metrics—namely, F1, accuracy (ACC), and area under the ROC curve (AUC)—and hierarchy-aware metrics—specifically, hF, hierarchical precision (hP), hierarchical recall (hR), and path-based loss. This enables direct analysis of structural prediction quality and hierarchical error propagation.

This work makes five primary contributions:

(1) **Unified HiGEC framework:** A modular framework integrating automatic hierarchy generation and hierarchy exploitation for flat multi-class classification is proposed.
(2) **Enhanced exploitation schemes:** Two novel HE variants, HE+ and HE+F, are introduced to mitigate hierarchical error propagation and improve prediction consistency through probabilistic path aggregation and flat–hierarchical fusion.
(3) **Hierarchy-aware evaluation:** In addition to conventional flat metrics, hierarchy-aware metrics are incorporated to quantify structural consistency and validate claims regarding error propagation mitigation.
(4) **Large-scale component analysis:** Extensive experiments systematically investigate the effects of HG methods, HE strategies, fusion mechanisms, and classifier selection across 100 datasets.
(5) **Open-source implementation and benchmark:** The complete HiGEC framework and benchmark pipeline are released as open-source software[1] (Alagoz 2025), enabling reproducible research in hierarchy-aware classification.

The remainder of this paper is organized as follows. Section 2 reviews the theoretical foundations of hierarchical classification and related HG approaches. Section 3 presents the proposed HiGEC framework and analyzes its computational complexity. Section 4 describes the experimental design and empirical results, including hierarchy-aware evaluation. Section 5 discusses the findings, limitations, and implications of the study. Finally, Section 6 concludes the paper.

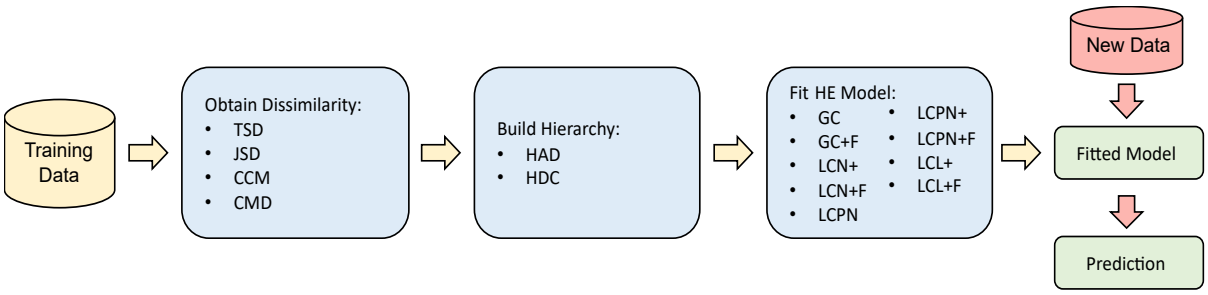


Fig. 1. HiGEC Framework: Training data is used to obtain dissimilarities, build a hierarchy, and fit an HC model. The fitted model can then be used to predict labels for new data points.

2 Background

This section establishes the technical and conceptual foundations necessary for understanding the proposed HiGEC framework. We begin by introducing the core principles of HC and its relationship to FC. Next, we describe the key mechanisms for obtaining class dissimilarities and constructing hierarchies from data. The section concludes with a review of related work, situating our contributions within the broader landscape of hierarchical and multi-class classification research.

[1] https://github.com/alagoz/higec

2.1 Hierarchical Classification

In traditional ML research, emphasis has been placed primarily on standard classification tasks, wherein each item is assigned to a single class selected from a set of mutually exclusive, non-overlapping categories. In the literature, classifiers that disregard the relationships among class labels have generally been referred to as "flat" classifiers. However, it has been recognized that this terminology may obscure important nuances within multi-class settings. For example, decomposition-based methods such as one-versus-all and all-versus-all, which do not utilize hierarchical structures, are also commonly categorized as flat classifiers.

In this study, a more precise definition has been adopted: FC, also known as single-classifier classification, is defined as the extension of a learning algorithm to directly manage more than two classes without employing decomposition strategies and without incorporating inter-class relationships into the optimization process.

In contrast, HC is characterized by the use of a predetermined hierarchical structure to guide classification outputs, rendering it a specific form of structured classification. Within this framework, class labels are arranged into a hierarchy, allowing an instance to be assigned to one or more categories either directly or through inherited relationships embedded within the hierarchical organization.

The following nomenclature is used throughout the paper. A dataset D containing m examples is defined as:

$$D = \{(x^{(0)}, y^{(0)}), (x^{(1)}, y^{(1)}), ..., (x^{(m-1)}, y^{(m-1)})\} \tag{1}$$

Here, $x^{(i)} \in \mathbb{R}^n$ is a feature vector with n-dimension that represents the ith example in the dataset. A multi-class dataset with k categories, without any hierarchical structure, is denoted by discrete variables $C = (c_0, c_1, ..., c_{k-1})$. When considering a general hierarchy built from flat labels, the class labels of the hierarchy are defined as:

$$\Gamma = \left(\gamma_0, \gamma_1, ..., \gamma_{k-1}, ..., \gamma_{\eta-1}\right) \tag{2}$$

Here, $c_j \triangleq \gamma_j$ for $c_j \in C$, and $\gamma_j \subseteq C$ for $j \in (k, .., \eta - 1)$. This representation ensures that hierarchical classes are composed of flat classes as well as their ancestor classes.

In FC, a classifier f maps each instance x to a class in C:

$$f : x \rightarrow C \tag{3}$$

Additionally, this mapping can produce probabilistic values for the classes.

In HC, the classification task can be delegated to different regions of the hierarchy. A classifier h assigns each instance x a subset of class values in Γ:

$$h : x \rightarrow \Omega_\Gamma \tag{4}$$

where Ω_Γ is a subset of Γ. The exact structure of Ω_Γ depends on the specific characteristics of the HC framework.

2.1.1 Hierarchy Framework Characterization. The hierarchical framework can be characterized based on three aspects, as outlined by Silla and Freitas (Silla and Freitas 2011):

- Structure: Defines the type of graph used, such as tree or directed acyclic graph (DAG).
- Labeling: Specifies whether paths are single-path (each instance is classified into exactly one class) or multi-path (each instance can be classified into multiple classes simultaneously).
- Depth: Indicates whether the hierarchy has full depth (all levels of the hierarchy are used) or partial depth (only a subset of levels is used).

In this study, we adopt a binary tree hierarchy with single-path labeling and full-depth prediction. For such a hierarchy, we define a class label notation that captures the position of each node, its parent, and its level within the binary structure. Each node label $\ell_{u,a}^{v,d}$ encodes the following information: u denotes the node index, a identifies its parent (ancestor) index, v corresponds to the hierarchical level (depth), and d indicates the branch

direction, where 0 denotes a left descendant and 1 denotes a right descendant. The complete set of node labels is expressed as:

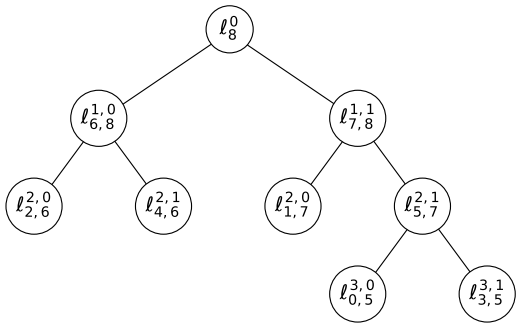
$$\Gamma = \left(\ell_{0,a_0}^{v_0,d_0}, \ell_{1,a_1}^{v_1,d_1}, \ldots, \ell_{u,a_u}^{v_u,d_u}, \ldots, \ell_{\eta-1,a_{\eta-1}}^{v_{\eta-1},d_{\eta-1}} \right), \tag{5}$$

where η denotes the total number of nodes in the binary hierarchy.

Figure 2 illustrates this labeling scheme on a trivial 5-class classification problem. In this hierarchical structure, singleton nodes represent terminal (leaf) classes corresponding to original class labels, while non-singleton nodes correspond to internal (non-leaf) categories formed by clustering leaf nodes. For a complete binary tree with k leaf classes (original categories), the total number of nodes η is given by $\eta = 2k - 1$, ensuring a full hierarchical representation encompassing both internal and terminal nodes.

Alternatively, the structure of the hierarchy can be captured in an explicit linkage table, as shown in the right panel of Figure 2. This table extends the standard linkage matrix provided by the SciPy library by explicitly encoding parent-child relationships and assigning meaningful node heights. In contrast to SciPy's convention—where node heights reflect clustering distances—heights in this representation are inversely aligned with hierarchical levels, such that nodes positioned higher in the hierarchy are associated with greater height values. This detailed representation enhances interpretability and enables precise control during hierarchy construction and exploitation phases.

a) Label Settings



b) Linkage Details

node_id	parent_id	child_ids	level_id
0	5	None	3
1	7	None	2
2	6	None	2
3	5	None	3
4	6	None	2
5	7	[0, 3]	2
6	8	[2, 4]	1
7	8	[1, 5]	1
8	None	[6, 7]	0

Fig. 2. Visualization of hierarchical label encoding and its corresponding linkage representation for a 5-class toy problem. **(a) Label Configuration**: A binary tree structure is used to hierarchically arrange the five flat classes. Each node is labeled as $\ell_{u,a}^{v,d}$, where u is the node index, a denotes its parent, v indicates the level in the hierarchy, and d specifies left (0) or right (1) child. Leaf nodes correspond to individual classes, while internal (non-leaf) nodes represent clusters formed during hierarchy construction. This encoding ensures both structural clarity and compatibility with different tree traversal algorithms employed in HE strategies. **(b) Linkage Table**: A detailed linkage table that extends the SciPy linkage matrix by explicitly encoding parent-child relationships and associating each node with a level-aware height. The hierarchical height aligns inversely with the depth level.

2.1.2 Types of Hierarchical Classifiers. There are four categories of algorithms that arrange classifier models in a hierarchical structure:

- Global Classifier (GC): A single global classifier is trained to predict all classes within the hierarchy.
- Local Classifier per Node (LCN): A binary classifier is trained for each node (excluding the root) to determine whether an instance is associated with that node's label. Different approaches exist to define the degree of

association, with training examples divided into positive and negative categories accordingly (Silla and Freitas 2011).
- Local Classifier per Parent Node (LCPN): A local classifier is trained for each parent node to predict its immediate child classes.
- Local Classifier per Level (LCL): A distinct classifier is trained for each level of the hierarchy, distinguishing among nodes within the same level.

This study introduces extensions and combinations for each of the HC schemes listed above.

2.2 Obtaining Dissimilarity

Constructing a class hierarchy necessitates an effective representation of class relationships. Since each class is defined by a set of examples, the method used to characterize and interpret these groups plays a pivotal role in shaping the resulting hierarchy. Several strategies have been developed for class representation, including centroid-based techniques such as Class Conditional Means (CCM), information-theoretic measures like Jensen-Shannon Divergence (JSD) and Task Similarity Distance (TSD), as well as classifier-driven approaches that leverage confusion matrix-derived distances (CMD).

2.2.1 CCM. The centroid-based method defines classes in the feature space directly using abstraction techniques and mathematical summarization. Determining class centers is one of the basic uses of this method. Calculating the CCM is a popular technique for identifying class centers. The CCM ($\mu_j \in \mathbb{R}^n$) for class c_j is determined by the following equation and is the component-wise average of all samples in that class:

$$\mu_j = \frac{1}{|D_j|} \sum_{x \in D_j} x \tag{6}$$

In this case, $|D_j|$ indicates how many examples are in class c_j. μ_j is the arithmetic mean of all the components of feature vectors in class c_j.

2.2.2 JSD. JSD calculates the difference between two probability distributions (Lin 1991). Some of the shortcomings of the Kullback-Leibler Divergence (KLD), such as asymmetry and sensitivity to sparse data, are addressed by this symmetric and smoothed variant of the KLD (Kullback and Leibler 1951). It is widely used in bioinformatics, ML, and information retrieval applications such as text classification, document clustering, and probability distribution comparison.

The JSD between two probability distributions, P and Q, is calculated as follows:

$$JSD(P, Q) = \frac{1}{2}(D_{KL}(P||M) + D_{KL}(Q||M)) \tag{7}$$

where $D_{KL}(P||M)$ denotes the KLD between distributions P and M where M is the average of P and Q:

$$M = \frac{1}{2}(P + Q) \tag{8}$$

The JSD is bounded between 0 and 1, where 0 indicates identical distributions and 1 signifies completely dissimilar distributions. JSD can be generalized to compare more than two classes/ distributions, such as:

$$\mathrm{JSD}_{\pi_1,\ldots,\pi_n}(P_1, P_2, \ldots, P_n) = \sum_i \pi_i D_{KL}(P \parallel M) = H(M) - \sum_{i=1}^{n} \pi_i H(\pi_i) \tag{9}$$

where $M := \sum_{i=1}^{n}$ and $\pi_1, ..., \pi_c$ are weights for probability distributions and $H(.)$ is the Shannon entropy.

In this study, JSD is computed for all pairs of classes, resulting in a pairwise dissimilarity matrix.

2.2.3 TSD. TSD, like JSD, is an information-theoretic metric that assesses how similar or different classes are from one another using the conditional distributions. But it also takes into account the classifiers' prediction behavior when they are used to differentiate between two groups. Consequently, TSD incorporates aspects of both classification-based and information-based representation learning.

Unlike JSD, which employs a reference distribution that is the average of the compared distributions, advancements in transfer learning (Pan and Q. Yang 2009) have inspired alternative approaches. These include using classification distributions as reference distributions (Gao and Chaudhari 2021; Helm, W. Yang, et al. 2021). These novel measures create two artificial classification distributions and introduce a reference conditional distribution to concentrate only on conditional distributions. To determine hierarchical structures, TSD uses task similarity (Helm, W. Yang, et al. 2021) as a metric.

In particular, as defined by (Helm, Mehta, et al. 2020), a third distribution M is introduced along with two classification distributions P' and Q' when utilizing task similarity. The similarity between two conditional distributions P and Q is thus measured. The following is a definition of these distributions:

$$P' = p(X|Y\neg P)P + (1 - p(X|Y\neg P))M \tag{10}$$

$$Q' = p(X|Y\neg Q)Q + (1 - p(X|Y\neg Q))M \tag{11}$$

The similarity is then calculated as:

$$s(P, Q) = \frac{1}{2}\left(TS(P', Q') + TS(Q', P')\right) \tag{12}$$

M is randomly selected from the collection of conditional distributions, excluding P and Q, in situations involving large-scale multi-class problems. A pairwise similarity matrix is then created by randomly selecting many instances of M_i and calculating the average similarity $s(P, Q)$. As a result, similarity becomes a stochastic metric. It is important to note that $s(P, Q)$ is scaled and asymmetrical. As a result, symmetry is obtained while building the dissimilarity matrix by averaging it with its transpose and normalizing all values to lie between 0 and 1. Finally, by deducting non-diagonal values from 1, similarity is converted to dissimilarity.

2.2.4 CMD. It is often assumed that two classes are similar if a classifier has difficulty differentiating between them, whereas easy differentiation indicates that they are different. One common way to assess a classifier's ability to discriminate between two classes is to evaluate its performance. It is possible to get supervised dissimilarity measures between classes by applying a classifier to the training data.

Analyzing each class pair's separability is one method. However, when there are a lot of classes, this might become computationally costly. A confusion matrix produced by a single classifier is frequently used by practitioners as a more effective solution to this problem. The rows of the confusion matrix can be used as class representatives (Bengio et al. 2010; Godbole 2002). Another method, proposed by Silva-Palacios et al. (Silva-Palacios et al. 2018), calculates the dissimilarity between two classes based on the accuracy derived from the relevant subset of the confusion matrix M:

$$M = \begin{bmatrix} m_{11} & \cdots & m_{1c} \\ \vdots & \ddots & \vdots \\ m_{c1} & \cdots & m_{cc} \end{bmatrix} \tag{13}$$

Using the confusion matrix elements, dissimilarity between two classes can be obtained as:

$$d(c_i, c_j) = \frac{m_{ii} + m_{jj}}{m_{ij} + m_{ji} + m_{ii} + m_{jj}} \tag{14}$$

This approach produces a pairwise dissimilarity matrix, where each entry quantifies the dissimilarity between two classes. In this study, the confusion matrix is derived using FC with a single split on the training set, allocating 80% of the data to the training-validation set.

2.3 Building Hierarchy

The process of building a hierarchy structure involves identifying the nodes and their connections. The input may consist of observation points in $\mathbb{R}^n$ or a pairwise dissimilarity matrix. Hierarchical Agglomerative Clustering (HAC) and Hierarchical Divisive Clustering (HDC) are two different clustering techniques that can be used to create a hierarchy.

2.3.1 HAC. HAC, sometimes referred to as the bottom-up technique, builds the hierarchy from the bottom up by progressively integrating discrete elements that begin as distinct clusters. The `linkage` function of the SciPy library, which takes as inputs either a distance matrix or observation vectors, is used to implement this technique. A variety of distance metrics are offered by `linkage` to assess the dissimilarity across clusters. Ward's distance metric is used in this investigation.

2.3.2 HDC. HDC, on the other hand, takes a top-down approach, beginning with a single cluster that contains every object and dividing it repeatedly to create the hierarchy. For matrix inputs, the K-Medoids algorithm (Kaufman and Rousseeuw 2009) is employed as the default clustering method. When the input consists of point cloud data, the K-Means algorithm is used. A custom algorithm for HDC has been developed as part of this study. The `linkHDC` function is used to create the hierarchy. It allows users to customize the hierarchy-building process to meet their individual needs by supporting both matrix and observation vector inputs and offering the option to split using a preferred clustering function.

2.4 Related Work

This study aims to enhance multi-class classification performance by unifying HC with automated HG. It spans three core areas of research: the design of HE strategies, the generation of data-driven class hierarchies, and the systematic evaluation of these approaches for performance improvement. By benchmarking 100 datasets, this work not only surveys existing HC and HG methods but also introduces and evaluates novel enhancements to HC.

The growing interest in hierarchical approaches is largely motivated by the challenges inherent in traditional multi-class classification. Common decomposition methods, such as OVA and AVA, become computationally expensive with an increasing number of classes. OVA requires k classifiers and AVA requires $k(k-1)/2$ classifiers for k classes. In contrast, hierarchical structures reduce the required number of classifiers to $O(\log_b k)$ for a balanced tree of branching factor b, offering computational scalability. Prior studies have demonstrated the efficiency and effectiveness of hierarchical methods over OVA and AVA (Frank and Kramer 2004; Vural and Dy 2004). Additionally, HC naturally captures label correlations, balances class distributions, and provides improved interpretability in large-scale problems.

However, manually designing semantic hierarchies is time-consuming and does not always lead to better classification outcomes (Lei et al. 2014; M. Sun et al. 2013). Modifications to predefined hierarchies have been shown to improve performance (Babbar et al. 2016; X. Li et al. 2021; Naik and Rangwala 2019), highlighting the need for automated HG. Despite this, the broader potential of HG for enhancing classification performance remains underexplored.

Efforts to automatically derive hierarchies from flat-labeled data generally fall into two categories: representative-based and classifier-based HG methods (Del Moral, Nowaczyk, Sant'Anna, et al. 2023). Representative-based approaches assess class dissimilarity using centroid distances (T. Li et al. 2007; Punera et al. 2005; Vural and Dy 2004) or distributional metrics such as Fisher projections (Y. Chen et al. 2004), JSD (Punera et al. 2006), and TSD (Helm, W. Yang, et al. 2021). These methods model classes as probabilistic distributions and compute inter-class dissimilarities accordingly. While interpretable and efficient, they may struggle with complex or noisy class boundaries.

Classifier-based HG methods, in contrast, use confusion matrices or classifier outputs to infer class relationships (Bengio et al. 2010; Godbole 2002; Silva-Palacios et al. 2018). Distances between confusion matrix rows or prediction distributions form the basis of hierarchy construction, often refined via semi-metric or spectral clustering techniques. These approaches are adaptive to classifier behavior but may depend heavily on the initial model's reliability.

Hierarchical clustering is widely employed in HG, most commonly through HAC and HDC. HAC follows a bottom-up approach using linkage methods such as Ward's method (Godbole 2002) or UPGMA (T. Li et al. 2007), while HDC takes a top-down approach using K-means (Vural and Dy 2004), Gaussian Mixture Models (Helm, W. Yang, et al. 2021), or graph-based strategies (Bengio et al. 2010). Some studies use heuristic or randomized optimization techniques to construct or refine hierarchies based on estimated classification performance (Frank and Kramer 2004; Melnikov and Hüllermeier 2018).

Exploiting the constructed hierarchy effectively—referred to as HE—is central to HC. A prominent strategy is the top-down classification approach, typically formalized as LCPN. Although the term LCPN was introduced more recently (Silla and Freitas 2011), its principles have been widely applied in earlier work (Bengio et al. 2010; Y. Chen et al. 2004; Cheong et al. 2004; Frank and Kramer 2004; Godbole 2002; Helm, W. Yang, et al. 2021; T. Li et al. 2007; Punera et al. 2005; Silva-Palacios et al. 2018), including in large-scale applications (Qu et al. 2016). LCPN operates via greedy path traversal, predicting a single path from root to leaf, but suffers from error propagation. To address this, probabilistic chain-rule-based methods were proposed, allowing for the combination of outputs from all levels (Frank and Kramer 2004; Qu et al. 2016).

Recent advancements, such as the LCPN+F scheme (Alagoz 2024), integrate outputs with hierarchical predictions. This hybrid design mitigates error propagation while improving accuracy by leveraging the complementary strengths of both models. Performance and efficiency analyses have confirmed LCPN+F's superiority over traditional LCPN and other schemes. Building on this, the present study introduces generalized HE+ and HE+F strategies that extend this fusion approach to both global and local hierarchical settings. These innovations address key limitations of existing HE methods while preserving their benefits.

Despite extensive work on HC, few studies have conducted large-scale, systematic comparisons across HC schemes. Most evaluations have relied on small datasets or assumed predefined hierarchies (Del Moral, Nowaczyk, Sant'Anna, et al. 2023; Romero et al. 2022; Silla and Freitas 2011). This study fills this gap by evaluating a wide array of HC configurations across 100 datasets, providing detailed insights into how base classifiers interact with hierarchy design and exploitation strategies.

Finally, the goals of this research align with the broader objectives of boosting algorithms (T. Chen and Guestrin 2016; Freund, Schapire, et al. 1996; J. H. Friedman 2001), which aim to improve predictive performance by combining multiple models. Unlike boosting, which aggregates weak learners, HiGEC decomposes the output space through structured label hierarchies, enabling the model to exploit class relationships and improve both scalability and accuracy. This offers a novel and effective alternative to traditional ensemble-based methods for multi-class classification.

3 Methods

This section presents the proposed HiGEC framework in detail. We first introduce the extended hierarchy exploitation schemes that form the core of our approach, followed by a computational complexity analysis to characterize the efficiency of the proposed methods. Formal theoretical verification is provided to establish the soundness of the framework. We then describe the comprehensive benchmarking setup, including the dataset collection, base classifiers, and evaluation protocols used throughout the experimental study.

3.1 Extended HE Schemes

In global classification, all labels within the hierarchy are considered simultaneously by a single classifier. While this approach enables comprehensive exploitation of the complete hierarchical relationships, significant limitations have been identified regarding the capture of fine-grained distinctions between classes and the maintenance of system modularity. These limitations primarily stem from the substantial overlap in sample spaces across different hierarchical levels, which complicates the learning process and reduces classification precision.

In contrast, local classification strategies are employed at specific points within hierarchies, with classifiers being deployed at parent nodes, hierarchical levels, or individual nodes. Modularity is preserved and localized information is effectively leveraged through these methods. Nevertheless, error propagation remains a key disadvantage, where misclassifications at higher levels adversely influence predictions at subsequent levels.

To address these limitations while while leveraging the strengths of both global and local classification paradigms, extended HE schemes are proposed in this study. In the extended local classification configuration, when predictions account for all labels defined within the hierarchy, the scheme is referred to as **HE+**. Alternatively, when either global or local HE methods are combined with outputs from flat classifiers, the resulting approach is designated as **HE+F**. A visual representation of classifier-to-label mappings for different strategies under these schemes is presented in Figure 3.

3.1.1 Extended Hierarchy Exploitation (HE+). The HE+ framework enhances conventional hierarchical classification through probabilistic path evaluation, addressing the error propagation limitation inherent in greedy hierarchical approaches. Building upon established local classification paradigms (Frank and Kramer 2004; Qu et al. 2016), HE+ introduces two principal prediction modalities:

(1) Maximum Path Probability:

$$\hat{y} = \underset{c_j \in C}{\operatorname{argmax}} \prod_{u \in \mathcal{P}(c_j)} h_q^H(\ell_u|x) \tag{15}$$

(2) Average Path Probability:

$$\hat{y} = \underset{c_j \in C}{\operatorname{argmax}} \frac{1}{|\mathcal{P}(c_j)|} \sum_{u \in \mathcal{P}(c_j)} h_q^H(\ell_u|x) \tag{16}$$

where $\mathcal{P}(c_j)$ denotes the complete ancestral path to class c_j, h_q^H denotes the probabilistic output of hierarchical classifier (with $H \in \{N, P, L\}$ for LCN, LCPN, or LCL schemes, respectively and q is index of hierarchy unit -node, parent, or level-) for the label ℓ_u in the ancestral path.

LCN+ Scheme. This configuration trains $2k - 2$ binary classifiers using a modified version of the exclusive policy (Eisner et al. 2005), as shown in Figure 3(c). Unlike traditional implementations that consider only partial depth annotations, this version is compatible with full-depth label assignments. Positive examples are identified based on the most frequently occurring hierarchical label in each instance, while the rest are treated as negative samples. This strategy ensures semantic consistency by avoiding the counterintuitive assignment of descendant labels as negatives.

LCPN+ Scheme. As shown in Figure 3(e), this variant trains a multi-class classifier at each parent node, responsible for distinguishing between its two child nodes. A total of $k - 1$ classifiers are deployed, following the standard example selection policy where training instances are assigned to child nodes based on their most common label.

LCL+ Scheme. One classifier is trained per level, with the number of levels ranging from $\lceil \log_2 k \rceil$ to $k - 1$, depending on hierarchy balance (Figure 3(g)). Inconsistencies between level-wise predictions are avoided through

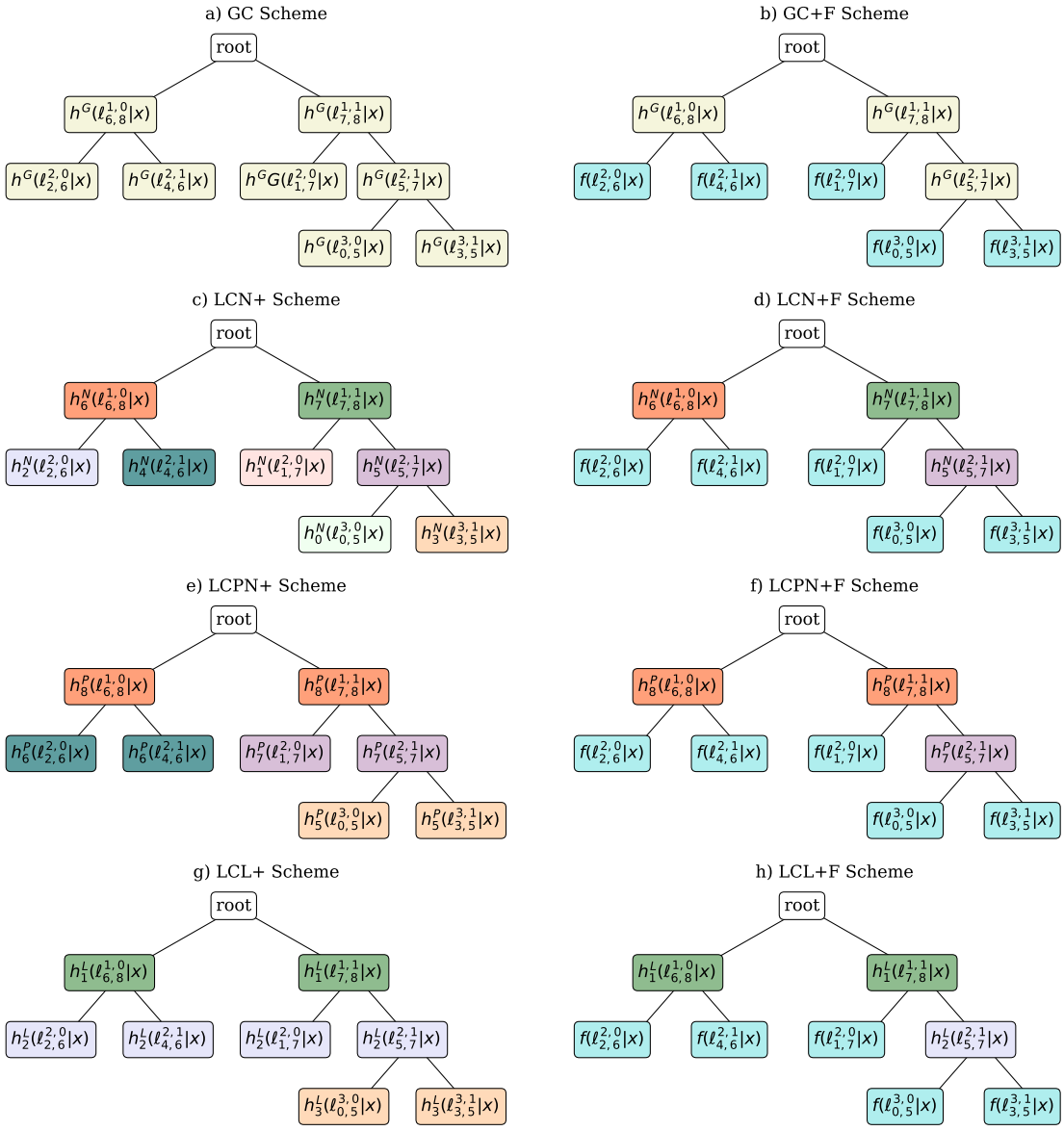


Fig. 3. Illustration of the HE+ and HE+F schemes applied to a hierarchy constructed for a 5-class toy problem, highlighting classifier-to-label associations by indicating corresponding probabilistic prediction outputs for each label. **HE+ Scheme (Left panels)**: A hierarchical prediction framework in which probabilities are computed along all root-to-leaf paths. Final predictions are derived either by maximizing the cascaded probabilities or by averaging them along each path. **HE+F Schemes (Right panels)**: An extended variant where predictions from flat classifiers are fused with hierarchical outputs to enhance prediction reliability and mitigate the impact of local errors. Color coding indicates classifier deployment locations (nodes/parents/levels) and their corresponding label prediction responsibilities, demonstrating different allocation strategies under each scheme. These visual cues demonstrate the classifier allocation strategies employed under each scheme.

the use of probabilistic path inference. This mitigates error propagation without enforcing rigid top-down constraints.

An overview of classifier characteristics under HE+ is provided in Table 1.

Table 1. HE+ Implementation Characteristics

Scheme	Classifiers	Notes
LCN+	$2k-2$ binary classifiers	One per non-root node
LCPN+	$k-1$ multi-class classifiers	One per parent node
LCL+	$[\lceil \log_2 k \rceil, k-1]$ classifiers	One per level

3.1.2 Hybrid Exploitation with Flat Classification (HE+F). The HE+F scheme integrates flat classifiers with hierarchical classifiers via a bypass mechanism. During inference, prediction for a given class c_j is made by combining the flat classifier output with the hierarchical path probability, defined as:

$$P(c_j|x) = f(c_j|x)^{\alpha} \prod_{u \in \mathcal{P}(c_j) \setminus c_j} h_q^H(\ell_u|x)^{1-\alpha} \tag{17}$$

The parameter $\alpha \in [0,1]$ governs the relative contribution of flat and hierarchical evidence. We fix $\alpha = 0.5$ as a neutral, non-informative prior assigning equal weight to flat and hierarchical components; tuning α is a promising direction for future work (Section 5.2).

Four HE+F variants demonstrate different forms of bypass and classifier allocation:

GC+F Scheme. A global classifier is trained on all non-leaf categories, while leaf-level predictions are handled exclusively by a flat classifier (Figure 3(b)). This partitioning reduces label overlap and model complexity during training and enhances decision stability through cascaded probability inference.

LCN+F Scheme. As illustrated in Figure 3(d), all leaf node classifiers are fully bypassed, and a flat classifier is used to distinguish among all k flat categories. The total number of classifiers becomes $1 + (k-2)$.

LCPN+F Scheme. In this variant (Figure 3(f)), parent classifiers are conditionally bypassed based on whether their child nodes are leaf categories. A full bypass occurs if both children are leaves, while a partial bypass is applied otherwise. The number of active classifiers typically ranges from $\lfloor \log_2 k \rfloor$ to $k-1$.

LCL+F Scheme. Shown in Figure 3(h), this scheme bypasses hierarchical classifiers at leaf-only levels entirely and applies partial bypass at levels containing both leaf and non-leaf nodes. The number of active classifiers also depends on tree depth and structure.

Implementation characteristics for HE+F variants are summarized in Table 2.

Table 2. HE+F Implementation Characteristics

Scheme	Classifiers	Bypass Condition
GC+F	1 FC + 1 HC	Partial (leaves only)
LCN+F	1 FC + $(k-2)$ HC	Full at leaves
LCPN+F	1 FC + $[\lceil \log_2 k \rceil, k-1]$ HC	Conditional
LCL+F	1 FC + $[\lceil \log_2 k \rceil, k-1]$ HC	Conditional

3.2 Computational Complexity Analysis

The computational requirements of the proposed hierarchical classification framework are examined in terms of time complexity, considering a dataset with m training instances, each represented by an n-dimensional feature vector. The depth of the hierarchy is denoted by d, which typically equals $\log_2 k$ in the case of balanced binary trees. The analysis is structured around three principal components: (i) HG, (ii) model training, and (iii) inference.

3.2.1 Hierarchy Generation Complexity. While HG methods may differ in specific steps, the overall time complexity is primarily dominated by the computation of the pairwise dissimilarity matrix. The complete complexity of the HG process can be expressed as:

$$\mathcal{T}_{\text{HG}} = O(m^2 n) + \mathcal{T}_{\text{Clustering}} \tag{18}$$

The first term corresponds to computing pairwise dissimilarities among all instances (each involving $O(n)$ operations), while the second term accounts for the cost of hierarchical clustering over k initial classes. Depending on the chosen method—HAC or HDC—the clustering complexity varies as follows:

$$\mathcal{T}_{\text{Clustering}} = \begin{cases} O(k^2(n + \log k)) & \text{(HAC)} \\ O(k^2 n \log k) & \text{(HDC, avg-case)} \end{cases} \tag{19}$$

In general, HDC exhibits higher computational complexity due to recursive subdivision and deeper tree construction.

3.2.2 Training Complexity. Training complexity is determined by the number of classifiers trained and the size of the data passed to each. These factors vary across the HE+ and HE+F schemes:

HE+ Variants. In the HE+ configuration, hierarchical classifiers are trained for every required unit of the tree. For the LCN+ scheme, binary classifiers are trained for each of the $2k - 2$ non-root nodes. In LCPN+, multi-class classifiers are assigned to each of the $k - 1$ parent nodes, with two child labels each. For LCL+, one multi-class classifier is trained per level, with the number of levels typically ranging from $\lceil \log_2 k \rceil$ to $k - 1$. The training complexity can be approximated by:

$$\mathcal{T}_{\text{HE+}} = \begin{cases} O((2k-2) \cdot \mathcal{T}_{\text{binary}}(m, n)) & \text{(LCN+)} \\ O((k-1) \cdot \mathcal{T}_{\text{multi}}(m', n, 2)) & \text{(LCPN+)} \\ O(d \cdot \mathcal{T}_{\text{multi}}(m', n, k')) & \text{(LCL+)} \end{cases} \tag{20}$$

Here, $\mathcal{T}_{\text{binary}}(m, n)$ refers to the cost of training a binary classifier, and $\mathcal{T}_{\text{multi}}(m, n, c)$ denotes the cost of training a multi-class classifier with c classes. The variable $m' \leq m$ accounts for the decreasing sample count at lower levels of the hierarchy, and $k' \leq k$ is the number of active classes per level. For the GC variant, where all nodes except the root are modeled jointly, the training complexity increases due to label overlap, approximated as:

$$\mathcal{T}_{\text{GC}} = O(\mathcal{T}_{\text{multi}}(md, n, 2k - 2))$$

HE+F Variants. In HE+F schemes, leaf-level predictions are delegated to a flat classifier, while only a subset of hierarchical classifiers are trained based on the structure of the tree and bypass conditions. For instance, in LCN+F, leaf node classifiers are excluded, resulting in $k - 2$ binary hierarchical classifiers in addition to one flat classifier. In LCPN+F and LCL+F, training bypasses are applied conditionally, depending on whether leaf classes are involved at each parent or level. The total training cost becomes:

$$\mathcal{T}_{\text{HE+F}} = \mathcal{T}'_{\text{HE+}} + \mathcal{T}_{\text{multi}}(m, n, k) \tag{21}$$

where $\mathcal{T}'_{\text{HE+}} \leq \mathcal{T}_{\text{HE+}}$, and the second term corresponds to the training of the flat classifier. For GC+F, the flat classifier replaces leaf labels entirely, and the global model is trained on non-leaf categories, leading to $O(\mathcal{T}_{\text{multi}}(m(d-1), n, k-2))$.

3.2.3 Inference Complexity. The inference cost per test instance is determined by the hierarchy structure and the evaluation strategy.

Flat Classifier (FC). A single model is evaluated, resulting in a complexity of $O(P(n))$, where $P(n)$ denotes the cost of making a prediction with an n-dimensional input.

Standard HE (Greedy). Only one root-to-leaf path is traversed, requiring d classifier evaluations, yielding $O(d \cdot P(n))$. Since paths must be determined for individual samples, the implementation requires sample-wise processing, leading to $O(d \cdot m \cdot P(n))$.

HE+ Schemes. All k root-to-leaf paths are evaluated in parallel, each involving d classifier calls, giving a complexity of $O(k \cdot d \cdot P(n))$. As sample-wise prediction is not required, batch processing (depending on classifier capabilities) can improve efficiency in practice.

HE+F Schemes. Due to classifier bypassing, only part of the hierarchy is traversed, and leaf predictions are delegated to a flat classifier. The average-case complexity becomes:

$$O\left(k \cdot d' \cdot P(n) + P_{\text{flat}}(n)\right),$$

where $d' \leq d$ and $P_{\text{flat}}(n)$ represents the flat classifier's prediction cost.

3.2.4 Summary of Complexity. The complexity characteristics of each scheme are summarized in Table 3, where C denotes a scheme-specific constant. Empirical runtimes may vary due to factors such as multiprocessing, and whether processing is performed sample-wise or in batches.

Table 3. Computational Complexity Summary for Hierarchical and Hybrid Schemes

Scheme	Training Complexity	Inference Complexity	Remarks
FC	$O(\mathcal{T}_{\text{multi}}(m, n, k))$	$O(P(n))$	No hierarchy used
Standard HE (Greedy)	$O(C \cdot \mathcal{T}(m/C, n))$	$O(d \cdot P(n))$	Risk of error propagation
HE+	$O(C \cdot \mathcal{T}(m/C, n))$	$O(k \cdot d \cdot P(n))$	Probabilistic, robust
HE+F	$\leq O(C \cdot \mathcal{T}(m/C, n) + \mathcal{T}_{\text{multi}}(m, n, k))$	$\leq O(k \cdot d \cdot P(n) + P_{\text{flat}}(n))$	Hybrid, more efficient
GC+F	$O(\mathcal{T}_{\text{multi}}(m(d-1), n, k-2))$	$\leq O(k \cdot d \cdot P(n) + P_{\text{flat}}(n))$	Compact, label overlap affects training

3.3 Theoretical Verification

To ensure the soundness and reproducibility of the proposed framework, we provide formal analyses of the extended hierarchy exploitation schemes in Section 6 in Appendix. Lemma A.1 establishes that the probabilistic averaging mechanism in HE+ reduces cumulative error propagation relative to greedy hierarchical inference. Proposition A.2 proves that the hybrid HE+F posterior remains probabilistically consistent when combining calibrated flat and hierarchical estimators, and Corollary A.3 extends this result to show that the hybrid posterior is Bayes-optimal under convex loss. These theoretical guarantees complement the empirical evaluation and provide a rigorous foundation for the observed improvements in predictive accuracy and hierarchical prediction consistency.

3.4 A Comprehensive Multiclass Classification Benchmark

To significantly expand the scope of multi-class classification evaluation, this study introduces a large-scale benchmark comprising 100 diverse datasets. This represents a substantial advancement over recent work such as (Del Moral, Nowaczyk, Sant'Anna, et al. 2023), which analyzed only 19 datasets, and surpasses the previously highest reported count of 27 datasets used in (Melnikov and Hüllermeier 2018). By incorporating a broader dataset collection, the study strengthens the empirical validity and generalizability of the proposed HiGEC framework.

All datasets were obtained from the OpenML repository and contain a mix of categorical (symbolic) and numerical features. Datasets on which any of the baseline classifiers achieved an F1 score greater than 0.99 were excluded to avoid performance ceilings and ensure meaningful evaluation.

Additionally, when multiple versions of the same dataset were available, only one version was retained to prevent redundancy, as minor variations typically do not produce significant differences in classification behavior. The final selection of datasets, along with key characteristics, is presented in Table 4. The number of class labels varies between 3 and 100 across the selected datasets.

The benchmark spans a wide array of domains, including biology, medicine, healthcare, life sciences, agriculture, planetary science, demographics, transportation, manufacturing, chemistry, geography, meteorology, sports, social media, and cultural analytics. All datasets are multivariate, including image-based data represented in multivariate feature form. Feature types include categorical, integer, and continuous values, reflecting the heterogeneous nature of real-world classification problems.

To enable hyperparameter analysis of HiGEC, a development set was constructed from 20% of the datasets in the benchmark. This subset was selected to preserve the overall class distribution profile of the full benchmark. The datasets comprising the development set are indicated with asterisks in Table 4.

3.5 Base Classifiers

To evaluate the robustness and generalizability of the HiGEC framework, a variety of widely used classification algorithms were selected as base learners. This not only facilitates the identification of high-performing models for multi-class classification but also allows the proposed framework to be evaluated across classifiers that vary in learning paradigms, levels of complexity, and inductive biases.

The selection was guided by three key criteria: (i) demonstrated effectiveness in multi-class settings, (ii) compatibility with tabular data comprising both numerical and categorical features, and (iii) representation of distinct algorithmic paradigms, including boosting, trees, generative models, instance-based learning, and deep learning.

Gradient Boosting Models: Three Gradient Boosting Decision Tree (GBDT) based classifiers were considered: XGBoost (`XGB`) (T. Chen and Guestrin 2016), CatBoost (`CATB`) (Prokhorenkova et al. 2018), and LightGBM (`LGB`) (Ke et al. 2017). These models were selected due to their efficiency and strong performance on structured data. `XGB` was utilized with regularization and a softmax objective for multi-class outputs. `CATB` was employed for its capability to handle categorical features without preprocessing. `LGB` was included for its histogram-based tree growth and memory efficiency.

Tree-Based Ensembles: Ensemble methods using decision trees were included to evaluate variance-reducing strategies. Random Forest (`RF`), Extremely Randomized Trees (`ETC`) (Geurts et al. 2006), and Classification and Regression Trees (`CART`) (Loh 2011) were selected. These models differ in how randomness and tree splits are introduced, providing a basis for comparing stability and interpretability.

Generative Models: Two generative approaches were employed: Linear Discriminant Analysis (`LDA`) and Gaussian Naive Bayes (`GNB`). LDA was used for its assumption of shared covariances between Gaussian-distributed classes, while GNB was selected due to its simplicity and conditional independence assumption.

Table 4. Multi-class benchmark. ID: OpenML dataset ID, n_{num}: Number of numerical features, n_{sym}: Number of symbolic features. * In the development set.

name	ID	k	m	$(n_{num} + n_{sym})$	name	ID	k	m	$(n_{num} + n_{sym})$
air-quality-and-pollution*	46880	4	5000	9 (9+0)	meta-batchincremental	276	3	71	62 (62+0)
alizadeh-2000-v2	46773	3	62	2093 (2093+0)	meta-instanceincremental	278	3	71	62 (62+0)
amazon-commerce-reviews*	1457	50	1500	10000 (10000+0)	mfeat-fourier	14	10	2000	76 (76+0)
analcatdata-halloffame	454	3	1340	15 (14+1)	mfeat-zernike*	22	10	2000	47 (47+0)
analcatdata-happiness	40709	3	60	3 (1+2)	mobile-price	46944	4	2000	20 (14+6)
anneal	2	5	898	9 (6+3)	okcupid-stem	45067	3	26677	13 (2+11)
arrythmia	5	9	412	278 (205+73)	olindda-outliers	42793	4	75	2 (2+0)
artificial-characters	1459	10	10218	7 (7+0)	ovarian-tumour	1086	3	283	54621 (54621+0)
auto-ml-selector	44200	5	96	15 (15+0)	page-blocks	30	5	5473	10 (10+0)
auto-univ-au4-2500	1548	3	2500	100 (58+42)	pasture	339	3	36	22 (21+1)
auto-univ-au7-1100	1552	5	1100	12 (8+4)	prnn-cushings*	457	3	25	2 (2+0)
bach*	4552	68	5586	16 (2+14)	re1-wc*	395	25	1657	3758 (3758+0)
baseball	185	3	1340	15 (14+1)	rmftsa-sleepdata	679	4	1024	2 (2+0)
bridges*	327	6	104	5 (1+4)	robot-failures-lp1	1516	4	88	90 (90+0)
calendar-dow	40663	5	399	32 (12+20)	robot-failures-lp2	1517	5	47	90 (90+0)
cars	455	3	406	5 (4+1)	rsctc2010-1	1077	3	105	22283 (22283+0)
cervical-cancer-risk	46829	5	858	6 (6+0)	rsctc2010-2	1078	3	105	22283 (22283+0)
cleveland-nominal	40711	5	303	7 (0+7)	rsctc2010-3	1079	5	95	22277 (22277+0)
cnae-9*	1468	9	1080	856 (856+0)	rsctc2010-4	1080	5	113	54675 (54675+0)
collins*	40971	30	1000	19 (19+0)	rsctc2010-6	1082	5	92	59004 (59004+0)
deng-reads	46786	9	264	22959 (22958+1)	sdss17	46955	3	78053	11 (9+2)
dgf	43035	5	698	39 (9+30)	segment	36	7	2310	19 (19+0)
diabetes-130-us	45069	3	101766	40 (11+29)	shuttle	40685	7	58000	9 (9+0)
diggle-table-a2	694	9	310	8 (8+0)	soybean*	42	19	682	3 (0+3)
eda-mortgage-ny	46480	6	87930	18 (18+0)	spectrometer	313	22	480	101 (100+1)
energy-efficiency	1472	30	750	9 (8+1)	students-academic-success	46960	3	4424	36 (19+17)
financial-risk-assessment	46515	3	15000	15 (15+0)	thyroid-allbp*	40474	5	2800	26 (6+20)
flags	285	6	186	28 (2+26)	thyroid-allhyper	40475	5	2800	26 (6+20)
flare	46184	6	1066	11 (0+11)	thyroid-allhypo	40476	5	2800	26 (6+20)
football-player-position	46764	4	3611	11 (11+0)	thyroid-allrep	40477	5	2800	26 (6+20)
golub-1999-v2	46780	3	72	1868 (1868+0)	thyroid-dis	40478	5	2800	26 (6+20)
hayes-roth*	329	3	160	4 (4+0)	tumors-11*	45084	11	174	12533 (12533+0)
heart-h*	1565	5	294	13 (13+0)	visualizing-livestock	685	5	130	2 (1+1)
heart-switzerland*	1513	5	123	12 (12+0)	volcanoes-a2	1528	5	1623	3 (3+0)
helena	41169	100	65196	27 (27+0)	volcanoes-a3	1529	5	1521	3 (3+0)
hepatitis-c	1087	3	283	54621 (54621+0)	volcanoes-b1	1531	5	10176	3 (3+0)
hypothyroid*	57	3	3769	22 (1+21)	volcanoes-b3	1533	5	10386	3 (3+0)
internet-firewall	46936	4	65532	11 (11+0)	volcanoes-b4	1534	5	10190	3 (3+0)
ipums-la-98-small	381	7	7485	43 (0+43)	volcanoes-b6	1536	5	10130	3 (3+0)
ipums-la-99-small	378	7	8844	42 (0+42)	volcanoes-d1	1538	5	8753	3 (3+0)
khan-2001	46781	4	83	1069 (1069+0)	volcanoes-d2	1539	5	9172	3 (3+0)
kropt*	184	18	28056	6 (0+6)	volcanoes-d3	1540	5	9285	3 (3+0)
led24	40677	10	3200	24 (0+24)	volcanoes-d4	1541	5	8654	3 (3+0)
led-7digit	40496	10	500	7 (7+0)	volcanoes-e1	1542	5	1183	3 (3+0)
leukemia	45092	3	68	7129 (7129+0)	volcanoes-e3	1544	5	1277	3 (3+0)
lung	45093	5	203	12600 (12600+0)	volcanoes-e5*	1546	5	1112	3 (3+0)
lymphoma-burkitt	1084	3	220	22283 (22283+0)	walking-activity*	1509	22	149332	4 (4+0)
mabbob-ela-as-2d	46264	5	1120	45 (45+0)	wine-quality-red	40691	6	1599	11 (11+0)
mental-health-detection	46879	4	540	14 (14+0)	wine-quality-white*	40498	7	4898	11 (11+0)
meta-all	275	3	63	62 (62+0)	zoo	62	6	97	16 (16+0)

Instance-Based Learning: To include non-parametric classification, a probabilistic extension of k-Nearest Neighbors was used. The Weighted Probabilistic KNN (`wpKNN`) method was implemented, where class probabilities were inferred from inverse-distance weighting of the k nearest neighbors (Dudani 1976). Neighbor searches were performed using `KDTree` or `BallTree`, depending on feature dimensionality. Predictions were normalized and optionally smoothed via softmax. Support for missing values and parallel inference was also integrated.

Deep Learning Method: A transformer-based neural model, FT-Transformer (`FTT`) (Gorishniy et al. 2021), was adopted to serve as a deep learning baseline. In this model, categorical and numerical features were embedded into a shared token space, followed by stacked attention layers. A special learnable token was used to aggregate global representations for final classification. For datasets with large numbers of numerical features, dimensionality reduction was applied using PCA, with an upper bound of 300 components.

Implementation Details: All models were implemented in Python. `XGB`, `LGB`, and `CATB` were accessed via their respective libraries, each configured with 100 estimators. `RF`, `ETC`, `CART`, `LDA`, and `GNB` were utilized from `scikit-learn`. The `FTT` model was implemented using PyTorch Lightning. Unless otherwise specified, default hyperparameters were retained across all classifiers.

3.6 Evaluation

The evaluation methodology is organized into three components. First, we define the evaluation metrics used to assess both flat and hierarchical classification performance. Second, we describe the evaluation protocol, including cross-validation procedures and runtime measurements. Third, we present the statistical comparison tools employed to assess the significance and consistency of the results.

3.6.1 Evaluation Metrics.

Flat Classification Metrics. The macro-averaged F1-score was adopted as the primary evaluation metric across all datasets and experimental settings. To provide additional performance perspectives, ACC and AUC were also included for comparative assessments involving selected HiGEC configurations against FC approaches.

The ACC metric represents the proportion of correctly classified instances among the total number of instances. While it offers a general sense of model correctness, it is known to be sensitive to class imbalance. In contrast, the macro-averaged F1-score integrates both precision and recall and computes the mean F1 across all classes, thereby mitigating bias toward dominant classes and offering a more balanced evaluation.

The AUC metric quantifies the ability of a classifier to rank positive instances higher than negative ones, and it is particularly informative in probabilistic models. For multi-class problems, the one-vs-rest strategy was applied to compute a macro-averaged AUC score. This metric has been widely adopted in performance evaluations, particularly when decision confidence and ranking quality are of interest.

Hierarchical Evaluation Metrics. To properly assess hierarchy exploitation methods—which fundamentally differ in how they traverse and aggregate hierarchical paths—flat metrics alone are insufficient. Following standard practice in hierarchical classification literature (Kiritchenko et al. 2005; Kosmopoulos et al. 2015; Silla and Freitas 2011), we employ hP, hR, hF, and path-based loss. These metrics capture the structural quality of predictions by rewarding predictions that are semantically close to true labels in the hierarchy, even when exact matches are incorrect.

Let C denote the set of all classes in the hierarchy, and for a predicted class $\hat{y}$ and true class y, let $\mathcal{P}(c)$ denote the ancestral path of class c including itself. The hierarchical precision, recall, and F-measure are defined as:

$$\mathrm{hP} = \frac{\sum_{i=1}^{n} |\mathcal{A}(\hat{y}_i, y_i)|}{\sum_{i=1}^{n} |\mathcal{P}(\hat{y}_i)|}, \tag{22}$$

$$\mathrm{hR} = \frac{\sum_{i=1}^{n} |\mathcal{A}(\hat{y}_i, y_i)|}{\sum_{i=1}^{n} |\mathcal{P}(y_i)|}, \tag{23}$$

$$\mathrm{hF} = \frac{2 \cdot \mathrm{hP} \cdot \mathrm{hR}}{\mathrm{hP} + \mathrm{hR}}, \tag{24}$$

where $\mathcal{A}(\hat{y}, y) = \mathcal{P}(\hat{y}) \cap \mathcal{P}(y)$ denotes the set of common ancestors. These metrics quantify partial correctness: a prediction that misclassifies a leaf node but correctly identifies its parent receives partial credit, unlike flat metrics which treat such a prediction as entirely incorrect.

Path-based loss measures the normalized hierarchical distance between predicted and true classes:

$$\ell_{\mathrm{path}}(\hat{y}, y) = \frac{|\mathcal{P}(\hat{y}) \setminus \mathcal{P}(y)| + |\mathcal{P}(y) \setminus \mathcal{P}(\hat{y})|}{|\mathcal{P}(y)| + |\mathcal{P}(\hat{y})|}. \tag{25}$$

This symmetric loss function ranges from 0 (identical paths) to 1 (completely disjoint paths). Unlike flat loss functions (e.g., 0-1 loss), path-based loss captures the severity of errors, providing a direct measure of error propagation in hierarchical prediction. Lower values indicate that predictions are hierarchically closer to true labels.

These hierarchical metrics are essential for empirically substantiating claims about error propagation mitigation and structural improvement, which flat metrics cannot capture.

3.6.2 Evaluation Protocol. All experiments were conducted using ten independent Monte Carlo cross-validation (MCCV) repetitions to ensure statistically sound performance estimates. In each repetition, 80% of the dataset was randomly selected for training, while the remaining 20% was used for inference. Hierarchies were generated exclusively using the training partitions within each run, potentially resulting in distinct hierarchical structures across repetitions.

For the CMD-based hierarchy generation approach, a validation subset comprising 25% of the training data was held out within each MCCV repetition to compute confusion matrices. Both training and inference durations (including the HG duration in case of HiGEC) were recorded for each run, and median values over the ten repetitions were reported to ensure robustness against outlier effects.

For hierarchical metrics, ancestral paths were precomputed for each hierarchy structure. Path-based loss and hierarchical precision/recall were computed using these precomputed paths to ensure computational efficiency. Statistical comparisons using hierarchical metrics employed the same protocol as flat metrics to maintain consistency.

3.6.3 Statistical Comparison Tools. To assess statistical significance and consistency of classifier performance across multiple datasets, two complementary comparison frameworks were employed: Critical Difference Diagrams (CDDs) and Multiple Comparison Matrices (MCMs).

Critical Difference Diagrams. CDDs were used to visualize average method rankings across datasets. Global differences among methods were first evaluated using the Friedman test (M. Friedman 1940). When the null hypothesis was rejected, post-hoc pairwise comparisons were conducted using two-tailed Wilcoxon signed-rank tests with Holm–Bonferroni correction to control the family-wise error rate under multiple comparisons (Benavoli et al. 2016; Garcia and Herrera 2008; Ismail Fawaz et al. 2019). Methods connected by horizontal bars in the diagrams indicate no statistically significant difference at $\alpha = 0.05$.

Both flat evaluation metrics (F1, ACC, AUC) and hierarchy-aware metrics (hP, hR, hF, path-based loss) were analyzed using the same statistical framework. For pairwise hierarchical metric comparisons reported outside the CDD analysis, two-tailed Wilcoxon signed-rank tests were additionally applied with significance threshold $p < 0.01$.

Multiple Comparison Matrix. To complement the ranking-based analysis, MCMs were employed to provide detailed pairwise comparisons between methods across datasets (Ismail-Fawaz et al. 2023). Each matrix cell reports the mean performance difference, win/tie/loss counts, and Wilcoxon signed-rank test significance for the corresponding pair of methods. This analysis enables identification of both the magnitude and direction of performance dominance while providing finer-grained interpretation than rank-based summaries alone. Holm–Bonferroni correction was again applied to account for multiple testing.

4 Experiments

This section evaluates the proposed HiGEC framework from four complementary perspectives: (i) predictive effectiveness relative to conventional FC; (ii) the influence of HG and HE components under various base classifier configurations; (iii) hierarchical prediction quality using structure-aware metrics; and (iv) computational efficiency and scalability.

The experiments were designed to answer the following research questions:

- **RQ1:** Can HiGEC consistently improve predictive performance over strong FC baselines?
- **RQ2:** Which HG and HE components contribute most to HiGEC's performance gains?
- **RQ3:** Do hierarchical methods produce structurally meaningful predictions beyond what flat accuracy metrics capture?
- **RQ4:** What computational trade-offs arise from hierarchical decomposition?

To address these questions, the evaluation proceeded in four stages. First, baseline performance of FC classifiers was established. Second, the effects of HG and HE parameters were analyzed across base classifier configurations. Third, a detailed runtime analysis characterized the computational efficiency of HiGEC components. Fourth, top-performing FC and HiGEC configurations were compared using ranking analysis, pairwise comparisons, and Pareto frontier visualization to assess predictive performance alongside efficiency-accuracy trade-offs.

4.1 Experimental Setup

All experiments were performed on a system equipped with an 11th Gen Intel(R) Core(TM) i7-11700 CPU (2.50 GHz, 16 cores) and 32 GB of RAM. Python was used for all implementations, and parallel processing was employed where applicable—particularly during the training of classifiers within hierarchical models—to enhance computational efficiency. Additionally, `FTT` and `CATB` were executed on an NVIDIA GeForce RTX 3060 GPU (1.78 GHz, 3584 cores) with 12 GB of dedicated memory.

The benchmark comprised 100 multi-class datasets retrieved from OpenML, as detailed in Section 3.4. Of these, 20 datasets were allocated to a development set, used primarily for hyperparameter analysis and model selection.

4.2 FC Performance of Base Classifiers

The performance of FC models using various base classifiers was evaluated on the development set. Figure 4 presents critical difference diagrams summarizing comparative results across F1, ACC, AUC, and runtime efficiency—where runtime encompasses both training and inference durations.

Ten classifiers were compared in total. Based on F1-score rankings, `XGB` (3.6), `RF` (4.2), `LGB` (4.4), `CATB` (4.4), and `ETC` (4.8) formed the top-performing clique. Conversely, `FTT` (8.7), `GNB` (7.3), and `wpKNN` (6.7) were positioned in the lowest-performing group. For ACC, the top-ranked classifiers were `CATB` (3.3), `RF` (3.4), `ETC` (4.0), `LGB` (4.2),

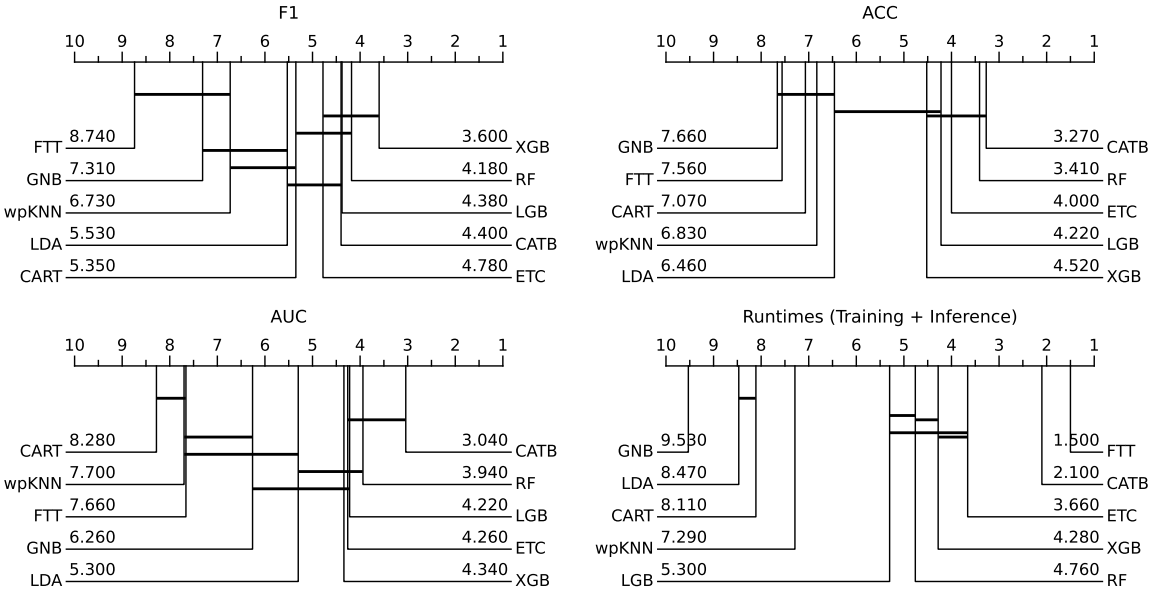


Fig. 4. Performance comparison of FC algorithms on the development set using critical difference diagrams. Metrics include F1-score, ACC, AUC, and total runtime (training + inference).

and XGB (4.5). The least effective classifiers in terms of ACC included GNB (7.7), FTT (7.6), CART (7.1), wpKNN (6.8), and LDA (6.5).

Regarding AUC performance, CATB (3.0), RF (3.9), LGB (4.2), and ETC (4.3) achieved the highest ranks, whereas CART (8.3), wpKNN (7.7), and FTT (7.7) performed poorly.

In terms of runtime efficiency, FTT (1.5) emerged as the slowest classifier, followed by CATB (2.1). The fastest models were GNB (9.6), LDA (8.5), CART (8.1), and wpKNN (7.3). Moderate runtimes were observed for ETC (3.7), XGB (4.3), RF (4.8), and LGB (4.8).

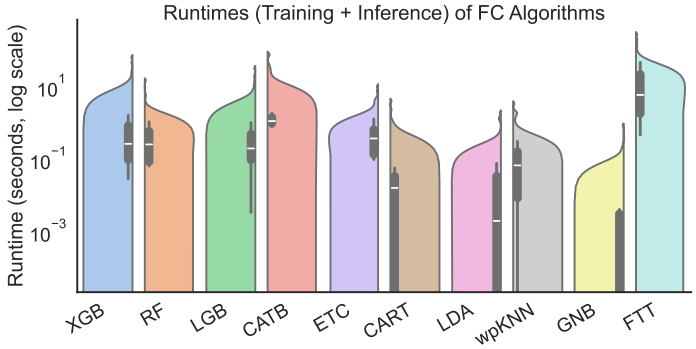


Fig. 5. Violin plot comparison of total runtimes (training + inference) for FC classifiers on the development set.

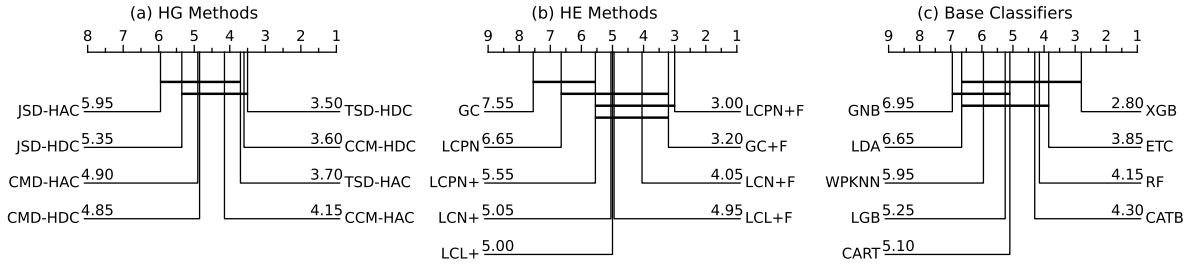


Fig. 6. CDDs for flat F-measure (F1) comparing (a) HG methods (dissimilarity-linkage combinations), (b) HE strategies, and (c) base classifiers. Methods connected by horizontal bars indicate no statistically significant difference ($\alpha = 0.05$, Wilcoxon-Holm post-hoc test). LCPN+F achieves the best average rank (3.0) among HE strategies, while CCM-HAC (4.15) leads among HG methods.

Figure 5 illustrates the runtime distributions of FC classifiers using violin plots. While the median runtimes of XGB, RF, and LGB are largely comparable, broader distribution tails are observed for XGB and LGB, indicating greater variability in execution time. This variability suggests higher sensitivity to dataset characteristics, particularly the product of the number of classes (k) and feature dimensionality. As detailed in Supplementary Material B (Alagoz 2025), datasets such as *ovarian-tumour* and *amazon-commerce-reviews* exhibit exceptionally high total runtimes for these classifiers, reinforcing this observation.

In summary, RF, XGB, CATB, LGB, and ETC demonstrated strong overall accuracy. Among these, CATB was the most computationally expensive. When speed is prioritized, LGB offers a favorable balance of efficiency and performance. In scenarios where accuracy is the primary concern, CATB is a suitable candidate. For a trade-off between speed and predictive power, RF and XGB emerge as ideal choices.

4.3 HiGEC Hyperparameter Analysis

Component Ranking Analysis. To systematically assess the contribution of HiGEC components, we performed a factor-wise ranking analysis by marginalizing over the remaining factors. Specifically, we evaluated (i) HG methods, (ii) HE strategies, and (iii) base classifiers independently. For each factor, performance was averaged across all combinations of the other components and evaluated over 20 held-out datasets. Statistical significance was assessed using the Friedman test followed by Wilcoxon-Holm post-hoc analysis ($\alpha = 0.05$), with results visualized using CDDs (Demšar 2006).

For flat F-measure (F1), statistically significant differences were observed across all three factors (HG: $p = 0.0084$, HE: $p < 10^{-6}$, classifiers: $p < 10^{-5}$). Among HG methods, TSD-based and CCM-based variants consistently achieved better average ranks, while JSD-based and CMD-based methods tended to rank lower. However, the CDD cliques indicated substantial overlap, suggesting that multiple HG configurations yield statistically comparable performance.

For HE strategies, clearer separation emerged. LCPN+F and GC+F achieved the best ranks, followed by LCN-based variants, while vanilla LCPN and GC consistently ranked lowest. This highlights the importance of feature-enhanced and calibrated ensemble variants. Similar trends were observed in classifier rankings, although differences were less pronounced due to overlapping cliques.

For hF, all factors again showed statistically significant differences (HG: $p = 0.0073$, HE: $p < 10^{-5}$, classifiers: $p < 10^{-8}$). Unlike F1, HG methods exhibited stronger overlap within a single clique, indicating that hierarchical performance is less sensitive to the choice of dissimilarity-linkage combination. In contrast, HE strategies

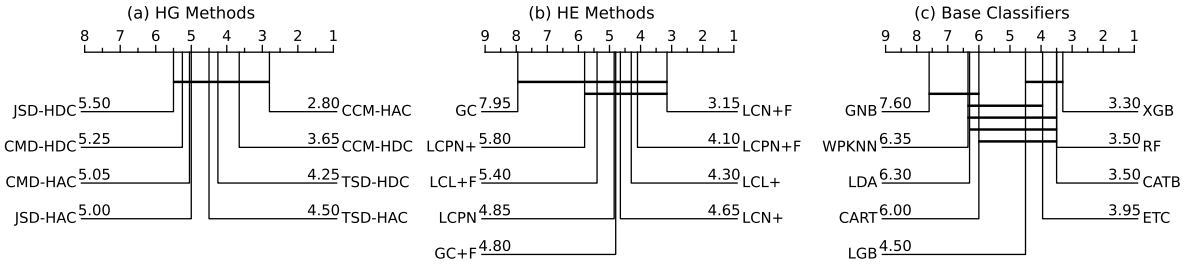


Fig. 7. CDDs for hF. LCN+F achieves the best average rank (3.15) among HE strategies, followed by LCPN+F (4.1). CCM−HAC ranks highest (2.8) among HG methods. Notably, performance differences among HG methods are less pronounced for hF compared to flat F1, suggesting that hierarchical evaluation is less sensitive to the choice of HG method.

remained influential, with LCN+F and LCPN+F achieving the best ranks. Notably, the baseline methods (e.g., GC) consistently underperformed.

Overall, the component ranking analysis revealed that: (1) HE strategies contribute the most to performance variation; (2) HG methods have moderate but consistent effects; (3) base classifiers show comparatively smaller differences under aggregation.

Cross-Classifier Consistency. Analysis of per-classifier results (see Supplementary Section S3) revealed consistent patterns across datasets. In particular, configurations combining CCM−HAC with feature-enhanced HE strategies (e.g., LCPN+F, LCN+F) frequently achieved top performance for both flat and hierarchical metrics. Strong learners such as XGB, RF, and CATB benefited most from HiGEC, showing consistent improvements in both accuracy and structural metrics, while weaker classifiers (e.g., LDA, GNB) exhibited limited gains. Additionally, improvements in path-based loss were consistently aligned with gains in hierarchical F-measure, reinforcing the effectiveness of hierarchy-aware optimization. These observations further support the global ranking results, highlighting HE strategy as the dominant factor and CCM-based HG methods as consistently effective choices across classifiers.

4.4 Evaluation Using Hierarchical Metrics

To complement flat evaluation metrics, we analyzed HiGEC using hierarchy-aware measures, including hF, hP, hR, and path-based loss. These metrics capture structural consistency in hierarchical predictions by rewarding predictions that are semantically close to true labels even when exact matches are incorrect.

Figure 8 presents a detailed comparison of LCPN, LCPN+, and LCPN+F using XGB as the base classifier aggregated across multiple HG configurations. The results demonstrate consistent improvements when moving from baseline to enhanced variants.

Hierarchical F-measure. LCPN+F achieved the highest hierarchical performance with mean hF of 0.9424 (±0.0505), outperforming LCPN (0.9321 ± 0.0642) and LCPN+ (0.9343 ± 0.0603). This corresponds to a 1.10% relative improvement over the LCPN baseline (Wilcoxon signed-rank test, $p = 0.0093$). Notably, all hierarchical F-measures substantially exceeded the flat classification baseline (mean F1 = 0.5749), with LCPN+F achieving a 63.9% relative improvement, underscoring the fundamental limitation of flat metrics.

Path-Based Loss. Path-based loss, which quantifies normalized hierarchical distance between predicted and true classes, decreased from 0.0626 (±0.0588) for LCPN to 0.0583 (±0.0511) for LCPN+F, representing a 6.88% reduction. This directly substantiates that the HE+F scheme mitigates error propagation, bringing predictions closer to true labels in the hierarchy.

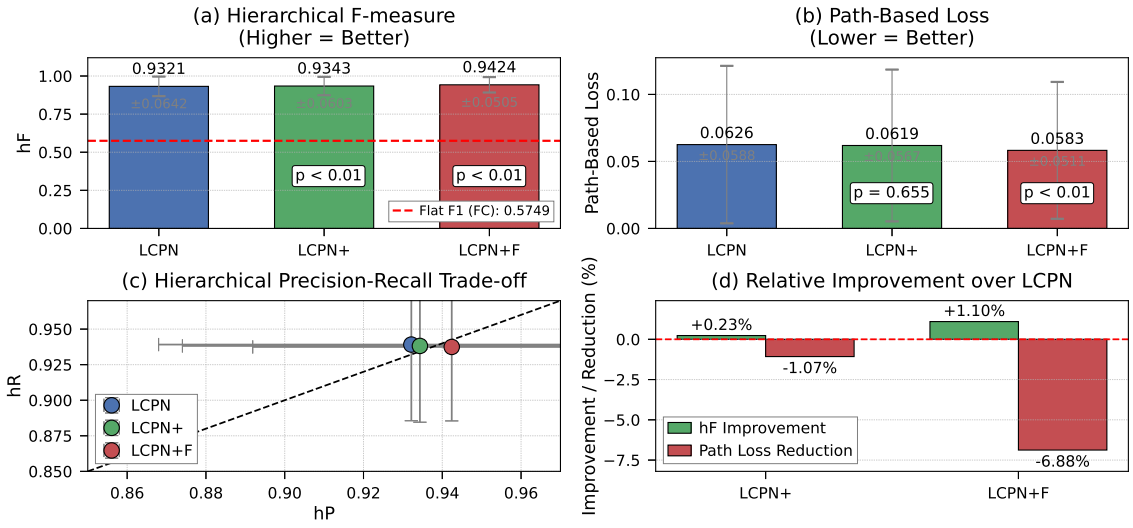


Fig. 8. Comparison of hierarchical evaluation metrics across LCPN, LCPN+, and LCPN+F using CCM-HAC generated hierarchies with XGB as the base classifier. (a) Hierarchical F-measure with flat F1 baseline (red dashed line) showing substantial gains from hierarchy exploitation. (b) Path-based loss quantifying hierarchical prediction error (lower is better). (c) Precision-recall trade-off demonstrating balanced performance across methods. (d) Relative improvements over LCPN baseline. Error bars indicate standard deviation across 20 held-out datasets. Statistical significance (Wilcoxon signed-rank test) is indicated for pairwise comparisons.

Precision-Recall Analysis. The precision-recall analysis (Figure 8c) shows that improvements are primarily driven by increased hierarchical precision (hP: 0.9321 → 0.9424), while hierarchical recall remains relatively stable (hR: 0.9391 → 0.9375). This suggests that enhanced variants reduce over-generalization errors without sacrificing coverage.

Statistical Validation. Wilcoxon signed-rank tests confirmed that improvements of LCPN+F over LCPN are statistically significant for both hF ($p = 0.0093$) and path-based loss ($p < 0.01$).

Summary of Hierarchical Evaluation. Hierarchical metrics collectively demonstrate that: (1) all methods produce structurally meaningful predictions (hF substantially exceeding flat F1); (2) LCPN+F consistently outperforms LCPN across all measures; (3) path-based loss reductions (6.88%) confirm error mitigation claims; (4) improvements are statistically significant ($p < 0.01$) and generalize across diverse datasets.

4.5 Empirical Runtime Analysis

Understanding computational cost is essential for selecting appropriate HiGEC configurations. While the FC framework comprises two stages—model fitting and inference—HiGEC introduces additional overhead through HG and HE, resulting in four stages: dissimilarity computation, hierarchy construction, model fitting, and prediction. Each component is analyzed below.

4.5.1 Hierarchy Generation Runtime. HG consists of dissimilarity computation and hierarchy construction. As hierarchy construction required negligible time, it is included within the dissimilarity computation stage. Runtimes were recorded during hyperparameter analysis on the development set (20 datasets).

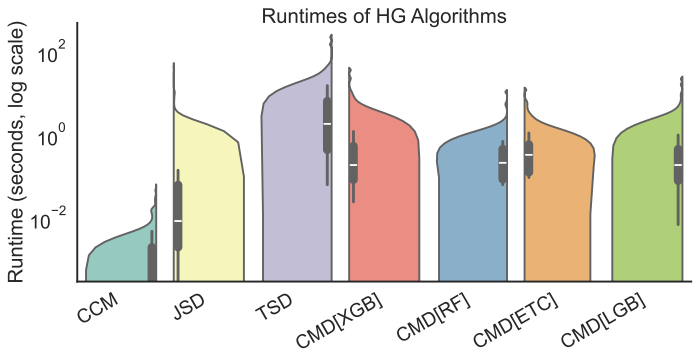


Fig. 9. Runtime distributions for HG dissimilarity measures across base classifiers.

Figure 9 summarizes runtime distributions for CCM, JSD, TSD, and CMD. Statistical measures (CCM, JSD) are substantially faster than classification-based measures (CMD, TSD). CCM exhibits the lowest runtime (median < 0.001 s), followed by JSD (median < 0.01 s). CMD requires moderate computation (0.18–0.33 s depending on classifier), while TSD is the most computationally expensive (median ≈ 1.8 s).
Observation: Statistical dissimilarity measures provide substantial computational advantages, making them suitable for large-scale or time-constrained scenarios.

4.5.2 Hierarchy Exploitation Runtime. HE runtime was evaluated for XGB, LGB, RF, and ETC. Figure 10 presents training and inference time distributions.

Fig. 10. Training and inference runtimes for HE schemes across classifiers (log-scale).

Training. LCPN, LCPN+, and LCL+ generally require less training time than FC, whereas fusion-based methods (e.g., LCPN+F, LCN+F, GC+F) incur additional cost due to training multiple models. Across classifiers, median training times typically follow XGB < LGB < RF < ETC.

Inference. FC provides the lowest inference time overall. Most HE schemes achieve comparable inference performance, with the exception of LCPN, which is slower due to sample-wise processing. Methods such as LCN+ exhibit inference times close to FC.

Training–Inference Asymmetry. In most configurations, inference is substantially faster than training (approximately 30× to 150×), indicating that HiGEC is well-suited for scenarios involving repeated prediction after a single training phase.

4.5.3 Impact of Multiprocessing. Experiments were conducted on a 16-core CPU with multiprocessing enabled for HE training. Figure 11 compares single- and multi-processing runtimes for XGB.

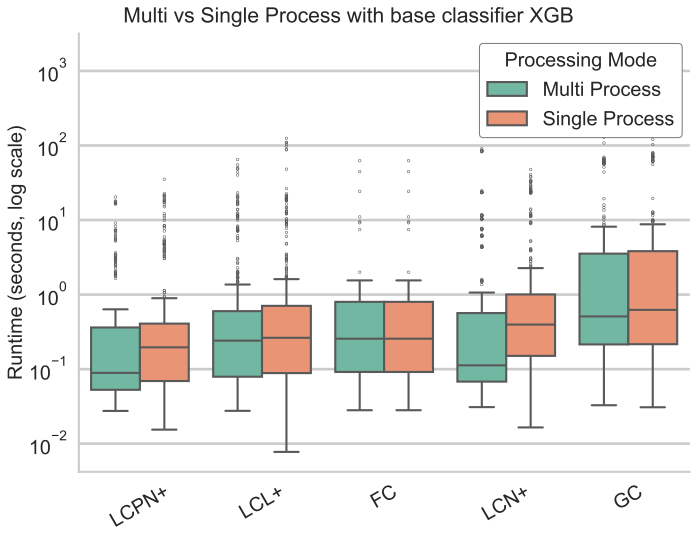


Fig. 11. Training runtime comparison between single- and multi-processing for XGB. Multiprocessing reduces training time by 30–72%, with greatest benefits for LCN+ (72%) and LCPN+ (55%).

Multiprocessing consistently reduces training time across all HE schemes. The largest improvements are observed for LCN+ and LCPN+, where median runtimes decrease by approximately 55–72%. Gains for simpler methods (e.g., LCL+, GC) are more modest.

4.5.4 Summary of Runtime Characteristics. Table 5 summarizes the key runtime properties of HiGEC components and representative configurations.

Key Findings:

(1) Statistical HG methods (CCM, JSD) provide substantial efficiency advantages;
(2) Fusion-based HE strategies improve flexibility at the cost of additional training time;
(3) Inference remains efficient for most HiGEC variants;
(4) Multiprocessing significantly mitigates training overhead, particularly for hierarchical methods.

These findings motivate the efficiency–accuracy trade-off analysis presented in Section 4.6.

Table 5. Summary of runtime characteristics (↓ lower is better).

Component/Config	Training	Inference	Parallel Gain
FC	Fast	Fastest	–
CCM / JSD (HG)	Very fast	–	–
CMD (HG)	Moderate	–	–
TSD (HG)	Slow	–	–
LCN+	Moderate	Near-FC	High (72%)
LCPN+	Fast	Near-FC	High (55%)
LCPN+F	Slow	Near-FC	Moderate
LCL+	Fast	Near-FC	Low (8%)
GC	Slow	Near-FC	Low (18%)
LCPN	Fast	Slow	–

4.6 Comparison of Selected HiGEC Schemes with FC

To investigate whether HiGEC provides practical advantages over competitive FC methods, we conducted a focused comparison between representative HiGEC configurations and the strongest FC baselines. All HiGEC schemes employed the CCM−HAC hierarchical grouping strategy, which demonstrated the best overall performance among evaluated HG methods.

Based on hyperparameter and runtime analyses, we selected four representative HiGEC configurations spanning the efficiency-accuracy spectrum: a high-efficiency non-fusion variant (LCN+[XGB]), a balanced configuration (LCPN+[XGB]), a high-accuracy fusion variant (LCPN+F[XGB]), and an alternative fusion configuration (LCPN+F[RF]). These were compared against top flat classifiers (XGB, RF, ETC, LGB) across 100 benchmark datasets using CDD, MCM, Pareto frontier analysis, and detailed pairwise comparisons.

4.6.1 Ranking Analysis. Figure 12 presents the CDD comparison. The Friedman test rejects the null hypothesis for F1 ($p = 0.00047$), ACC ($p = 6.47 \times 10^{-8}$), AUC ($p = 1.28 \times 10^{-5}$), and runtime ($p = 1.23 \times 10^{-73}$), confirming statistically significant differences among methods across all metrics.

For F1 score, LCPN+F[XGB] achieved the best average rank (3.68), followed by LCN+[XGB] (4.145) and XGB (4.32). LCPN+F[RF] (rank 4.5) trailed its XGB-based counterpart, suggesting that the flat classifier component strongly influences fusion performance.

For ACC, RF led (rank 3.65), while LCPN+F[XGB] remained competitive (4.42). For AUC, all HiGEC variants except LCN+[XGB] belonged to the top clique, indicating that fusion-based strategies maintain ranking quality comparable to FC.

For runtime, LCPN+[XGB] achieved the best rank (2.12), followed by LGB (2.83). Critically, LCN+[XGB] (3.98) ran faster than XGB (4.12), demonstrating that hierarchical decomposition can improve both accuracy and efficiency.

4.6.2 Pairwise Comparison Matrix. Table 6 provides granular pairwise significance testing. For F1, LCPN+F[XGB] significantly outperformed all FC baselines ($p < 0.005$), establishing it as the most highest-performing HiGEC configuration for F1 optimization. LCN+[XGB] showed a more nuanced pattern: it significantly outperformed ETC and LGB ($p < 0.05$) but not XGB or RF, despite a positive win rate (54 wins vs. 46 losses).

4.6.3 Efficiency-Accuracy Trade-off: Pareto Frontier. Figure 13 reveals the Pareto frontier. Five configurations are Pareto-efficient: ETC, LCPN+[XGB], RF, LCN+[XGB], and LCPN+F[XGB]. Critically, XGB is dominated—it achieves

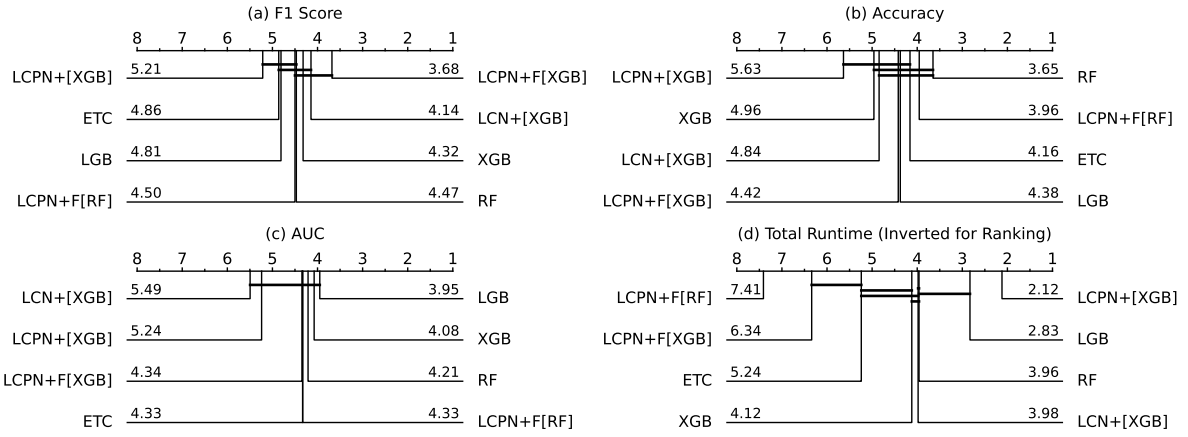


Fig. 12. Critical Difference Diagrams comparing selected HiGEC schemes with top flat classifiers across (a) F1 score, (b) accuracy, (c) AUC, and (d) total runtime. Methods connected by horizontal bars indicate no statistically significant difference ($\alpha = 0.05$, Wilcoxon-Holm post-hoc test). Average ranks are shown in parentheses. LCPN+F[XGB] achieves the best F1 rank (3.68), while LCN+[XGB] (3.98) runs faster than XGB (4.12).

	LCPN+F[XGB]	LCN+[XGB]	LCPN+F[RF]	LCPN+[XGB]
Mean-Accuracy	0.5735	0.5719	0.5609	0.5560
XGB 0.5686	**-0.0049** **31 / 2 / 66** **0.0001**	-0.0034 46 / 0 / 53 0.5275	0.0077 52 / 0 / 47 0.4003	**0.0125** **63 / 0 / 36** **0.0024**
RF 0.5553	**-0.0182** **39 / 0 / 60** **0.0037**	-0.0167 50 / 0 / 49 0.1713	-0.0056 51 / 2 / 46 0.6363	-0.0008 58 / 0 / 41 0.3867
ETC 0.5523	**-0.0212** **39 / 0 / 60** **0.0004**	**-0.0197** **43 / 0 / 56** **0.0398**	**-0.0086** **42 / 0 / 57** **0.0366**	-0.0037 53 / 0 / 46 0.8917
LGB 0.5362	**-0.0373** **38 / 0 / 61** **0.0015**	**-0.0358** **41 / 0 / 58** **0.0107**	-0.0247 49 / 0 / 50 0.7376	-0.0199 48 / 0 / 51 0.7908

Mean-Difference
r>c / r=c / r<c
Wilcoxon p-value

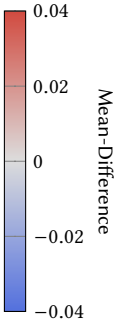


Table 6. MCM between selected HiGEC configurations and FC baselines using F1 score across 100 datasets. Each cell reports mean difference, win/draw/loss counts, and Wilcoxon signed-rank test p-value. Bold entries indicate statistically significant differences ($p < 0.05$)

lower F1 (0.5686) than LCN+[XGB] (0.5719) with higher runtime (4.02s vs. 3.01s). This Pareto dominance demonstrates that LCN+[XGB] is strictly superior to XGB on both objectives.

4.6.4 Pairwise Dataset-Level Analysis. Figure 14 presents per-dataset comparison between LCPN+F[XGB] and XGB. The configuration achieves 67 wins, 31 losses, and 2 ties (67% win rate), with mean F1 improvement of +0.91% ($p < 0.001$). The largest gains occur on challenging datasets such as robot-failures-lp2 (+9.5%) and volcanoes-d2 (+8.8%). Correlation analysis reveals no significant relationship between improvement and dataset characteristics (features: $r = 0.19$, $p = 0.06$; classes: $r = 0.05$, $p = 0.61$).

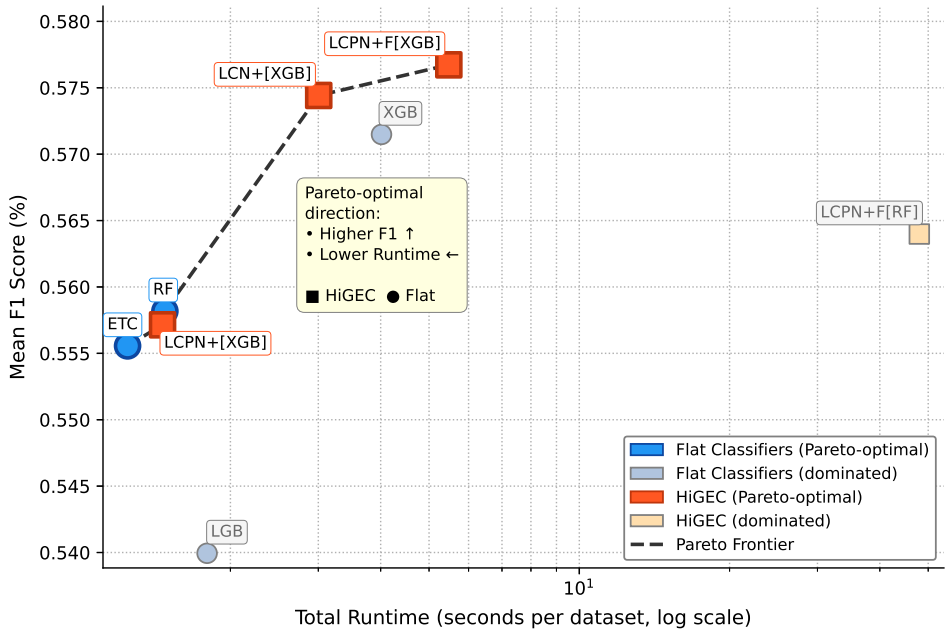


Fig. 13. Pareto frontier of efficiency-accuracy trade-off for selected HiGEC schemes and flat classifiers. XGB is dominated by LCN+[XGB] (higher F1, lower runtime). LCPN+F[XGB] achieves the highest F1 (0.5735) at greatest runtime (5.48s).

As supplementary analysis, LCN+[XGB] demonstrated a significant positive correlation with the number of features ($r = 0.32$, $p = 0.001$), indicating that lightweight hierarchical decomposition is particularly beneficial for high-dimensional problems. However, a negative correlation with the number of classes ($r = -0.22$, $p = 0.029$) suggests that benefits diminish with large label spaces.

4.6.5 Summary: Configuration Guidelines. Based on these analyses, we recommend:

- **For maximum accuracy (offline deployment):** LCPN+F[XGB] achieves the highest F1 (0.5735) with statistically significant improvements over all FC baselines ($p < 0.005$). Runtime overhead (5.48s) is acceptable for training-once, inference-often scenarios.
- **For efficiency-accuracy balance:** LCN+[XGB] Pareto-dominates XGB (higher F1, lower runtime) and shows strong improvements on high-dimensional datasets ($r = 0.32$, $p = 0.001$). This configuration is recommended for most practical applications.
- **For runtime-sensitive applications:** LCPN+[XGB] achieves the fastest runtime among HiGEC configurations (1.46s) while maintaining competitive F1 (0.5560).
- **For RF users:** LCPN+F[RF] provides an accuracy-focused alternative (mean F1=0.5609), though it underperforms compared to XGB-based fusion.

5 Discussion

This work introduces HiGEC, a unified framework integrating HG and HE for flat multi-class classification, and evaluates it through a large-scale benchmark of 100 tabular datasets. Beyond proposing new HE strategies, the study systematically analyzes the relative contributions of HG methods, HE schemes, and base classifiers through

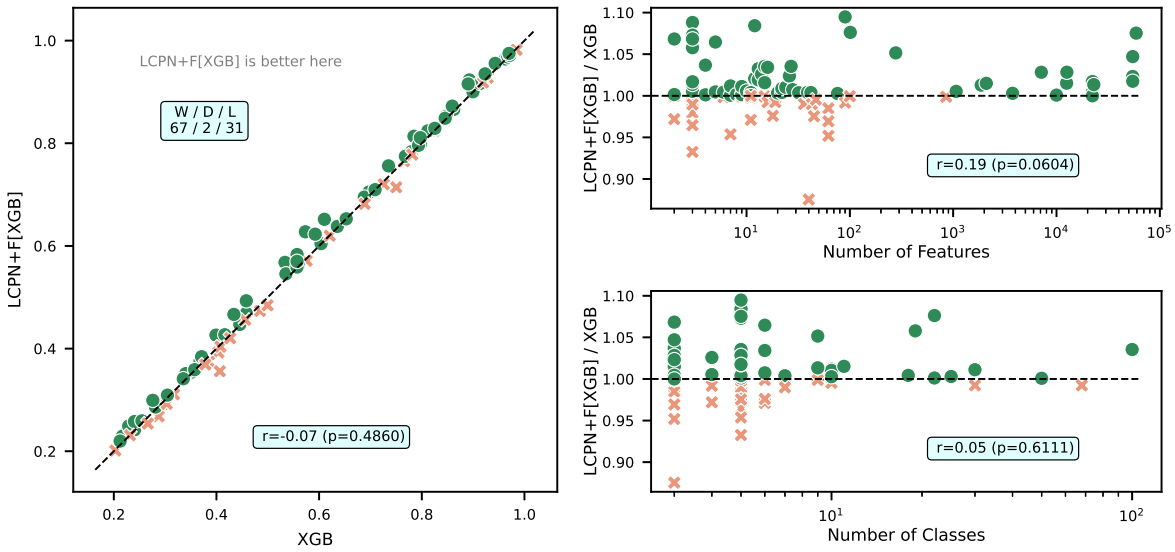


Fig. 14. Pairwise comparison between LCPN+F[XGB] and XGB across 100 datasets. (a) Per-dataset F1 scatter plot showing 67 wins, 31 losses, and 2 ties. (b) Relative improvement distribution. (c-d) Correlation analyses with dataset characteristics.

statistical ranking analysis, hierarchical evaluation metrics, runtime analysis, and pairwise comparisons against strong FC baselines.

5.1 Key Findings and Contributions

Effectiveness of Gradient Boosting Algorithms. The baseline FC benchmarking confirms trends previously observed in the tabular learning literature (Ayus et al. 2023; Bentéjac et al. 2021; Santoro et al. 2024; Wu et al. 2021). Gradient-boosted decision tree methods, particularly XGB, LGB, and CATB, consistently outperform alternative classifiers across heterogeneous datasets. Among these, XGB achieves the strongest balance between predictive performance and computational efficiency, making it particularly suitable as the backbone classifier for hierarchical decomposition.

Deep Learning Limitations on Tabular Data. Transformer-based and neural approaches showed comparatively weaker performance on the evaluated tabular benchmarks, consistent with prior findings (Ayus et al. 2023; Santoro et al. 2024; Wu et al. 2021). These observations reinforce the view that deep architectures remain less competitive than GBDT methods on structured tabular datasets characterized by moderate dimensionality and limited sample sizes. Consequently, the strongest HiGEC configurations were generally obtained using tree-based ensemble learners rather than neural architectures.

Statistical Gains from HiGEC.. The empirical results demonstrate that HiGEC can consistently improve predictive performance relative to FC, particularly for F1-score. Among the evaluated configurations, LCPN+F[XGB] achieved the strongest overall predictive performance and significantly outperformed all FC baselines in pairwise comparisons. Importantly, improvements were validated not only using flat metrics but also using hierarchical metrics, including hP, hR, hF, and path-based loss. The results indicate that HiGEC improves both predictive accuracy and structural consistency of predictions.

Addressing the Three Formal Hypotheses. The empirical findings support all three hypotheses introduced in Section 1. First, HiGEC consistently improved predictive performance over FC baselines, confirming that automatically generated hierarchies can provide measurable benefits on tabular multi-class datasets. Second, the proposed HE+ and HE+F strategies improved hierarchy-aware evaluation, indicating that the enhanced exploitation mechanisms preserve hierarchical structure more effectively than conventional HE approaches. Third, the observed reduction in path-based errors supports the claim that probabilistic path aggregation mitigates hierarchical error propagation. Together, these findings provide empirical support for the central design principles underlying the HiGEC framework.

Impact of Hyperparameters and Component Choices. The component ranking analysis revealed that HE strategies contribute most strongly to overall performance variation, whereas HG methods exert moderate but consistent effects across datasets. Among HG approaches, CCM-HAC consistently emerged as the strongest and most computationally efficient hierarchy-generation strategy. HE schemes incorporating probabilistic aggregation and feature-enhanced fusion (e.g., `LCPN+F` and `LCN+F`) consistently achieved superior ranks for both flat and hierarchical metrics. These findings suggest that the exploitation mechanism has greater influence on predictive quality than the specific hierarchy-generation method itself.

Isolating Component Contributions. To better attribute the observed performance gains, a factor-wise ranking analysis was conducted by marginalizing over the remaining experimental factors. The analysis indicates that the choice of HE strategy is the primary driver of predictive performance, whereas HG methods provide smaller but consistent improvements. Differences among base classifiers remain important, but their contribution is comparatively less influential once the effects of HG and HE are aggregated. These findings suggest that optimizing hierarchy exploitation has a greater impact on overall performance than refining hierarchy generation alone.

The Pareto frontier analysis provides additional insight by isolating the effect of hierarchical decomposition from flat–hierarchical fusion. The results indicate that hierarchy-aware decomposition alone can improve the trade-off between predictive performance and computational efficiency, demonstrating that the benefits of HiGEC are not solely attributable to fusion-based ensemble effects.

Dependence on Dataset Properties. Dataset-level analysis showed that HiGEC improvements were generally more pronounced on high-dimensional datasets. In particular, `LCN+[XGB]` exhibited statistically significant positive correlation between relative improvement and feature dimensionality. Conversely, weaker trends were observed with respect to the number of classes, suggesting that simpler local-classifier schemes may become less effective in extremely large label spaces. These observations indicate that hierarchical decomposition is especially beneficial for complex feature spaces rather than merely large class counts.

Computational Considerations. Runtime analysis demonstrated that computational overhead depends strongly on both HG and HE design choices. Statistical HG methods such as CCM and JSD were substantially faster than classification-based methods such as CMD and TSD. Among HE schemes, lightweight hierarchical approaches such as `LCN+[XGB]` and `LCPN+[XGB]` offered favorable efficiency–accuracy trade-offs, whereas fusion-based approaches incurred additional training cost due to combined flat and hierarchical prediction stages. Nevertheless, multiprocessing substantially reduced training times, particularly for decomposition-based methods, improving scalability for large-scale applications.

Statistical Depth. All statistical analyses employed the Friedman test to assess global differences, followed by Wilcoxon signed-rank post-hoc tests with Holm-Bonferroni correction ($\alpha = 0.05$). Holm correction was applied within each family of pairwise comparisons to control the family-wise error rate across multiple experimental conditions. In addition to critical difference diagrams, pairwise mean comparison matrices and Pareto frontier

analysis were used to provide complementary perspectives on statistical significance, performance consistency across datasets, and efficiency–accuracy trade-offs.

Generalization Scope. The empirical evaluation is restricted to tabular datasets obtained from OpenML. Accordingly, claims of general applicability are intentionally limited to hierarchy-aware learning on tabular benchmarks. Validation on domains with predefined semantic hierarchies (e.g., ImageNet, biomedical ontologies, or text taxonomies), as well as systematic analysis of robustness to label noise, class imbalance, and label ambiguity, remains an important direction for future work.

Relationship to Ensemble Learning. HiGEC shares conceptual similarities with ensemble learning because hierarchical decomposition combines multiple local decision functions. However, several important distinctions exist. First, boosting primarily operates through iterative reweighting in feature space, whereas HiGEC constructs structure in label space. Second, boosting aggregates weak learners additively, while HE+F combines flat and hierarchical predictions using calibrated convex fusion. Third, HiGEC's path-based aggregation explicitly exploits hierarchical label dependencies, which are typically not modeled in conventional flat ensemble methods. Nevertheless, formal connections between hierarchical exploitation and ensemble theory remain an important direction for future investigation.

Overall Interpretation. Collectively, the findings suggest that HiGEC provides the greatest benefit in high-dimensional multi-class problems where predictive performance is prioritized over minimal training cost. The results further indicate that hierarchy-aware decomposition can function not merely as an auxiliary ensemble mechanism, but as a meaningful structural learning strategy capable of improving both predictive accuracy and hierarchical consistency when appropriately optimized.

5.2 Limitations and Future Work

Limitations. Several limitations should be acknowledged. First, the current HG process generates static hierarchies prior to classifier training. Adaptive or dynamically evolving hierarchies may better capture changing class relationships during learning. Second, the empirical analysis focuses exclusively on tabular OpenML benchmarks, and therefore conclusions cannot yet be generalized to domains such as text classification, image recognition, or genomics without further validation. Third, although multiprocessing significantly mitigates runtime overhead, computational costs associated with hierarchy generation and multi-path inference may still limit deployment in strict real-time environments.

Calibration of the Fusion Parameter. The fusion parameter α in HE+F was fixed at 0.5 throughout the experiments to ensure consistent comparison across configurations. While this neutral setting simplifies benchmarking, the optimal fusion balance likely depends on dataset characteristics, hierarchy quality, and classifier behavior. Adaptive calibration of α through cross-validation, Bayesian optimization, or meta-learning may further improve predictive performance and prediction stability across datasets.

Future Research Directions. Several directions emerge for future investigation:

(1) **Adaptive HG and HE mechanisms**: Developing hierarchy-generation and exploitation strategies that automatically adapt to dataset characteristics such as imbalance, dimensionality, or sparsity.
(2) **Online and incremental HiGEC**: Extending the framework to streaming and dynamically evolving datasets through online hierarchy updates.
(3) **Domain-aware hierarchy generation**: Incorporating expert-defined semantic priors such as biomedical ontologies, taxonomies, or image hierarchies into HG.
(4) **Instance-level adaptive fusion**: Dynamically weighting flat and hierarchical predictions using uncertainty estimation or confidence calibration.

(5) **Adaptive classifier selection**: Exploring heterogeneous combinations of local and flat classifiers through meta-learning or optimization-based approaches.

Broader Applicability. Despite its current evaluation scope, HiGEC's modular architecture allows integration of both generated and predefined hierarchies, making it adaptable to a broad range of hierarchy-aware classification settings. The framework therefore provides a flexible empirical and methodological foundation for future research on hierarchical learning for structured multi-class problems.

6 Conclusions

This study introduced HiGEC, a unified framework that systematically integrates HG and HE for flat multi-class classification problems without predefined class structures. Through extensive evaluation on 100 benchmark datasets, HiGEC consistently improved predictive performance over conventional FC, with the strongest gains observed in F1-score and hierarchical evaluation metrics.

The proposed HE+ and HE+F schemes demonstrated that probabilistic path aggregation and flat–hierarchical fusion can mitigate hierarchical error propagation and improve the structural consistency of predictions. Hierarchical evaluation further confirmed these findings: `LCPN+F` achieved the highest hierarchical F-measure while reducing path-based loss relative to baseline hierarchical schemes, supporting the effectiveness of hierarchy-aware prediction aggregation.

Component-wise analyses revealed that HE strategies contributed most strongly to performance variation, while HG methods provided moderate but consistent gains. Among HG approaches, CCM-HAC emerged as the most most consistently high-performing and computationally efficient hierarchy generation strategy across classifiers. Pareto frontier analysis further showed that hierarchical decomposition can improve both predictive accuracy and runtime efficiency simultaneously. In particular, `LCN+[XGB]` achieved higher F1-score and lower runtime than the corresponding FC baseline `XGB`, demonstrating that hierarchy-aware decomposition may provide benefits beyond simple ensemble effects.

Runtime analyses demonstrated that optimized HE+F configurations typically incur moderate computational overhead—approximately 2–3× slower than FC—while achieving significantly improved predictive performance. At the same time, lightweight HE+ variants maintained competitive efficiency, indicating that HiGEC can support both accuracy-oriented and runtime-sensitive applications.

Statistical analyses using Friedman and Wilcoxon-Holm procedures confirmed that the observed improvements were statistically significant across diverse datasets. The large-scale benchmark, combined with factor-wise ranking analysis and pairwise comparisons, provides one of the most comprehensive empirical evaluations to date for hierarchy-aware classification on tabular multi-class datasets.

The empirical findings support the three formal hypotheses introduced in this study: (1) HiGEC improves flat predictive performance relative to FC baselines, (2) enhanced HE strategies improve hierarchy-aware evaluation metrics, and (3) probabilistic path aggregation mitigates hierarchical error propagation. Furthermore, factor-wise ranking analysis demonstrated that HE strategies contribute more strongly to performance variation than HG methods under aggregated evaluation.

Several limitations should be acknowledged. All experiments were conducted exclusively on tabular datasets obtained from OpenML, and therefore the findings cannot yet be generalized to structured domains such as image, text, or genomic classification. In addition, the effects of label ambiguity, class imbalance, and noisy hierarchy generation were not systematically investigated. Consequently, claims regarding the broader applicability of HiGEC are currently limited to the tabular benchmark setting evaluated in this work.

Overall, HiGEC establishes a unified methodological and empirical foundation for hierarchy-aware learning on flat multi-class tabular problems. The framework provides both a practical classification methodology and a reproducible benchmark platform for future research. Future work should investigate adaptive hierarchy

generation, instance-aware fusion strategies, and validation on semantic hierarchies and structured domains such as ImageNet and large-scale text taxonomies.

The complete implementation, experimental configurations, and evaluation scripts are publicly available at: https://github.com/alagoz/higec.

Acknowledgments

Funding in direct support of this work: Scientific and Technological Research Council of Turkey with grant number 125E949. The author has no relevant financial or non-financial interests to disclose.

References

C. Alagoz. 2024. "Hierarchy Generation and Exploitation via Extended Local Classification Schemes." In: *Proceedings of the 2024 16th International Conference on Machine Learning and Computing*, 462–469.

C. Alagoz. 2025. *HiGEC: Hierarchy Generation and Extended Classification Framework*. Available at: https://github.com/alagoz/higec. (2025).

M. Aly. 2005. "Survey on multiclass classification methods." *Neural Netw*, 19, 1-9, 2.

I. Ayus, N. Natarajan, and D. Gupta. 2023. "Comparison of machine learning and deep learning techniques for the prediction of air pollution: a case study from China." *Asian Journal of Atmospheric Environment*, 17, 1, 4.

R. Babbar, I. Partalas, E. Gaussier, M.-R. Amini, and C. Amblard. 2016. "Learning taxonomy adaptation in large-scale classification." *The Journal of Machine Learning Research*, 17, 1, 3350–3386.

A. Benavoli, G. Corani, and F. Mangili. 2016. "Should we really use post-hoc tests based on mean-ranks?" *The Journal of Machine Learning Research*, 17, 1, 152–161.

S. Bengio, J. Weston, and D. Grangier. 2010. "Label embedding trees for large multi-class tasks." *Advances in neural information processing systems*, 23.

C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz. 2021. "A comparative analysis of gradient boosting algorithms." *Artificial Intelligence Review*, 54, 1937–1967.

T. Chen and C. Guestrin. 2016. "Xgboost: A scalable tree boosting system." In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Y. Chen, M. M. Crawford, and J. Ghosh. 2004. "Integrating support vector machines in a hierarchical output space decomposition framework." In: *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*. Vol. 2. IEEE, 949–952.

S. Cheong, S. H. Oh, and S.-Y. Lee. 2004. "Support vector machines with binary tree architecture for multi-class classification." *Neural Information Processing-Letters and Reviews*, 2, 3, 47–51.

P. Del Moral, S. Nowaczyk, and S. Pashami. 2022. "Why is multiclass classification hard?" *IEEE Access*, 10, 80448–80462.

P. Del Moral, S. Nowaczyk, A. Sant'Anna, and S. Pashami. 2023. "Pitfalls of assessing extracted hierarchies for multi-class classification." *Pattern Recognition*, 136, 109225.

J. Demšar. 2006. "Statistical comparisons of classifiers over multiple data sets." *Journal of Machine learning research*, 7, Jan, 1–30.

S. A. Dudani. 1976. "The distance-weighted k-nearest-neighbor rule." *IEEE Transactions on Systems, Man, and Cybernetics*, 4, 325–327.

R. Eisner, B. Poulin, D. Szafron, P. Lu, and R. Greiner. 2005. "Improving protein function prediction using the hierarchical structure of the gene ontology." In: *2005 IEEE symposium on computational intelligence in bioinformatics and computational biology*. IEEE, 1–10.

E. Frank and S. Kramer. 2004. "Ensembles of nested dichotomies for multi-class problems." In: *Proceedings of the twenty-first international conference on Machine learning*, 39.

Y. Freund, R. E. Schapire, et al.. 1996. "Experiments with a new boosting algorithm." In: *icml*. Vol. 96. Citeseer, 148–156.

J. H. Friedman. 2001. "Greedy function approximation: a gradient boosting machine." *Annals of statistics*, 1189–1232.

M. Friedman. 1940. "A comparison of alternative tests of significance for the problem of m rankings." *The annals of mathematical statistics*, 11, 1, 86–92.

Y. Gao and P. Chaudhari. 2021. "An information-geometric distance on the space of tasks." In: *International Conference on Machine Learning*. PMLR, 3553–3563.

S. Garcia and F. Herrera. 2008. "An Extension on" Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons." *Journal of machine learning research*, 9, 12.

P. Geurts, D. Ernst, and L. Wehenkel. 2006. "Extremely randomized trees." *Machine learning*, 63, 3–42.

S. Godbole. 2002. "Exploiting confusion matrices for automatic generation of topic hierarchies and scaling up multi-way classifiers." *Annual Progress Report, Indian Institute of Technology–Bombay, India*.

Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. 2021. "Revisiting deep learning models for tabular data." *Advances in neural information processing systems*, 34, 18932–18943.

H. S. Helm, R. D. Mehta, B. Duderstadt, W. Yang, C. M. White, A. Geisa, J. T. Vogelstein, and C. E. Priebe. 2020. "A partition-based similarity for classification distributions." *arXiv preprint arXiv:2011.06557*.

H. S. Helm, W. Yang, S. Bharadwaj, K. Lytvynets, O. Riva, C. White, A. Geisa, and C. E. Priebe. 2021. "Inducing a hierarchy for multi-class classification problems." *arXiv preprint arXiv:2102.10263*.

B. W.-Y. Hsu and V. S. Tseng. 2024. "LightDPH: Lightweight Dual-Projection-Head Hierarchical Contrastive Learning for Skin Lesion Classification." *Journal of Healthcare Informatics Research*, 1–21.

H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. 2019. "Deep learning for time series classification: a review." *Data mining and knowledge discovery*, 33, 4, 917–963.

A. Ismail-Fawaz et al.. 2023. "An Approach To Multiple Comparison Benchmark Evaluations That Is Stable Under Manipulation Of The Comparate Set." *arXiv preprint arXiv:2305.11921*.

L. Kaufman and P. J. Rousseeuw. 2009. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. 2017. "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in neural information processing systems*, 30.

S. Kiritchenko, S. Matwin, A. F. Famili, et al.. 2005. "Functional annotation of genes using hierarchical text categorization." In: *Proc. of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*. Vol. 76.

A. Kosmopoulos, I. Partalas, E. Gaussier, G. Paliouras, and I. Androutsopoulos. 2015. "Evaluation measures for hierarchical classification: a unified view and novel approaches." *Data Mining and Knowledge Discovery*, 29, 3, 820–865.

S. Kullback and R. A. Leibler. 1951. "On information and sufficiency." *The annals of mathematical statistics*, 22, 1, 79–86.

H. Lei, K. Mei, N. Zheng, P. Dong, N. Zhou, and J. Fan. 2014. "Learning group-based dictionaries for discriminative image representation." *Pattern Recognition*, 47, 2, 899–913.

T. Li, S. Zhu, and M. Ogihara. 2007. "Hierarchical document classification using automatically generated hierarchy." *Journal of Intelligent Information Systems*, 29, 211–230.

X. Li, Y. Zhou, Y. Zhou, and W. Wang. 2021. "MMF: multi-task multi-structure fusion for hierarchical image classification." In: *International Conference on Artificial Neural Networks*. Springer, 61–73.

J. Lin. 1991. "Divergence measures based on the Shannon entropy." *IEEE Transactions on Information theory*, 37, 1, 145–151.

X. Liu and L. Wang. 2024. "Multi-granularity sequence generation for hierarchical image classification." *Computational Visual Media*, 10, 2, 243–260.

W.-Y. Loh. 2011. "Classification and regression trees." *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1, 1, 14–23.

V. Melnikov and E. Hüllermeier. 2018. "On the effectiveness of heuristics for learning nested dichotomies: an empirical analysis." *Machine Learning*, 107, 1537–1560.

A. Naik and H. Rangwala. 2019. "Improving large-scale hierarchical classification by rewiring: a data-driven filter based approach." *Journal of Intelligent Information Systems*, 52, 141–164.

A. Onan. 2023. "Hierarchical graph-based text classification framework with contextual node embedding and BERT-based dynamic fusion." *Journal of king saud university-computer and information sciences*, 35, 7, 101610.

S. J. Pan and Q. Yang. 2009. "A survey on transfer learning." *IEEE Transactions on knowledge and data engineering*, 22, 10, 1345–1359.

J. Peng, A. Ran, C. Yu, and H. Liu. 2024. "EGCNet: a hierarchical graph convolutional neural network for improved classification of electrocardiograms." *EURASIP Journal on Advances in Signal Processing*, 2024, 1, 93.

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. 2018. "CatBoost: unbiased boosting with categorical features." *Advances in neural information processing systems*, 31.

K. Punera, S. Rajan, and J. Ghosh. 2006. "Automatic construction of n-ary tree based taxonomies." In: *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*. IEEE, 75–79.

K. Punera, S. Rajan, and J. Ghosh. 2005. "Automatically learning document taxonomies for hierarchical classification." In: *Special interest tracks and posters of the 14th international conference on World Wide Web*, 1010–1011.

Y. Qu, L. Lin, F. Shen, C. Lu, Y. Wu, Y. Xie, and D. Tao. 2016. "Joint hierarchical category structure learning and large-scale image classification." *IEEE Transactions on Image Processing*, 26, 9, 4331–4346.

M. Romero, O. Ramírez, J. Finke, and C. Rocha. 2022. "Feature extraction with spectral clustering for gene function prediction using hierarchical multi-label classification." *Applied Network Science*, 7, 1, 28. doi:https://doi.org/10.1007/s41109-022-00468-w.

D. Santoro, T. Ciano, and M. Ferrara. 2024. "A comparison between machine and deep learning models on high stationarity data." *Scientific Reports*, 14, 1, 19409.

M. Shi, R. Wang, and J. Ren. 2023. "Hierarchical capsule network for hyperspectral image classification." *Neural Computing and Applications*, 35, 25, 18417–18443.

C. N. Silla and A. A. Freitas. 2011. "A survey of hierarchical classification across different application domains." *Data mining and knowledge discovery*, 22, 31–72.

D. Silva-Palacios, C. Ferri, and M. J. Ramírez-Quintana. 2018. "Probabilistic class hierarchies for multiclass classification." *Journal of computational science*, 26, 254–263.

A. Sun and E.-P. Lim. 2001. "Hierarchical text classification and evaluation." In: *Proceedings 2001 IEEE International Conference on Data Mining*. IEEE, 521–528.

M. Sun, W. Huang, and S. Savarese. 2013. "Find the best path: An efficient and accurate classifier for image hierarchies." In: *Proceedings of the IEEE International Conference on Computer Vision*, 265–272.

V. Vural and J. G. Dy. 2004. "A hierarchical method for multi-class support vector machines." In: *Proceedings of the twenty-first international conference on Machine learning*, 105.

J. Wu, Y. Li, and Y. Ma. 2021. "Comparison of XGBoost and the neural network model on the class-balanced datasets." In: *2021 IEEE 3rd international conference on frontiers technology of information and computer (ICFTIC)*. IEEE, 457–461.

B. Zeng, Y. Peng, J. Yang, P. Hong, and J. Liang. 2025. "CPM-based Hierarchical Text Classification." *Journal of Artificial Intelligence Research*, 82, 367–388.

A Proofs of Theoretical Properties

This appendix provides the formal proofs supporting the theoretical claims introduced in Section 3.3. Specifically, we demonstrate the error propagation mitigation of the proposed HE+ scheme and establish the probabilistic consistency and Bayes-optimality of the hybrid HE+F formulation. These derivations substantiate the theoretical soundness of our unified hierarchical–flat modeling approach.

A.1 Proof of Lemma 1: Error Propagation Mitigation

Lemma A.1 (Error Propagation Mitigation). *The HE+ scheme with average path probability reduces error propagation compared to greedy hierarchical methods. For a hierarchy with depth d, the probability of correct classification under error rate ϵ per level satisfies:*

$$P_{correct}^{HE+} \geq 1 - d\epsilon + O(\epsilon^2)$$

compared to $P_{correct}^{greedy} \leq (1-\epsilon)^d$ for greedy approaches.

Proof. Let X_i denote the correctness indicator at level i, where

$$X_i = \begin{cases} 1, & \text{if the prediction at level } i \text{ is correct,} \\ 0, & \text{otherwise,} \end{cases} \qquad \mathbb{E}[X_i] = 1 - \epsilon.$$

For greedy hierarchical inference, correct prediction requires success at all levels:

$$P_{\text{correct}}^{\text{greedy}} = \Pr\left(\bigwedge_{i=1}^{d} X_i = 1\right) = \prod_{i=1}^{d}(1-\epsilon) = (1-\epsilon)^d.$$

In contrast, HE+ aggregates probabilistic evidence along the entire ancestral path. Define the averaged correctness statistic:

$$\bar{X} = \frac{1}{d}\sum_{i=1}^{d} X_i,$$

with $\mathbb{E}[\bar{X}] = 1 - \epsilon$ and $\text{Var}(\bar{X}) = \frac{\epsilon(1-\epsilon)}{d}$ under independence. A sample is correctly classified if the averaged evidence exceeds a decision threshold τ (nominally 0.5). Applying Chebyshev's inequality,

$$\Pr(|\bar{X} - (1-\epsilon)| \geq t) \leq \frac{\epsilon(1-\epsilon)}{dt^2}.$$

Setting $t = (1-\epsilon) - \tau = 0.5 - \epsilon$ yields

$$\Pr(\bar{X} \leq \tau) \leq \frac{\epsilon(1-\epsilon)}{d(0.5-\epsilon)^2}.$$

Hence,

$$P_{\text{correct}}^{\text{HE+}} = 1 - \Pr(\bar{X} \le \tau) \ge 1 - \frac{\epsilon(1-\epsilon)}{d(0.5-\epsilon)^2}.$$

Expanding for small ϵ,

$$P_{\text{correct}}^{\text{HE+}} \ge 1 - d\epsilon + O(\epsilon^2),$$

indicating that the linear term in ϵ is equivalent to that of the greedy case, while the higher–order correction term is strictly smaller because of the $1/d$ averaging factor. Thus, the HE+ scheme exhibits lower expected cumulative error, particularly for deeper hierarchies where variance reduction through averaging becomes significant. □

A.2 Proof of Proposition 1: Probabilistic Consistency and Bayes-Optimality of HE+F

The HE+F scheme combines the predictions of a flat classifier $f(c_j|x)$ and a hierarchical classifier $h(c_j|x)$ into a single posterior:

$$P(c_j|x) = f(c_j|x)^{\alpha}\, h(c_j|x)^{1-\alpha}, \quad \alpha \in [0,1].$$

Taking logarithms, we obtain the convex combination in log space:

$$\log P(c_j|x) = \alpha \log f(c_j|x) + (1-\alpha) \log h(c_j|x).$$

Proposition A.2 (Probabilistic Consistency of HE+F). *If f and h are both calibrated probabilistic estimators minimizing expected log-loss, then $P(c_j|x)$ defined above is also calibrated and minimizes a convex combination of the respective risks.*

Proof. Let $\mathcal{L}(p, y) = -\log p(y|x)$ denote the log-loss. The expected risk of the hybrid model is:

$$R(P) = \alpha R(f) + (1-\alpha) R(h),$$

where $R(f) = \mathbb{E}_{x,y}[\mathcal{L}(f(c_j|x), y)]$ and similarly for $R(h)$. By convexity of log-loss, the risk of the hybrid model satisfies:

$$R(P) \le \alpha R(f) + (1-\alpha) R(h),$$

with equality if and only if $f = h$. Therefore, the fused distribution $P(c_j|x)$ remains a proper probabilistic predictor that preserves calibration and bounded expected loss. □

Corollary A.3 (Bayes-Optimality of Hybrid Posterior). *Given calibrated probabilistic estimators $f(c_j|x)$ and $h(c_j|x)$ minimizing expected log-loss, the hybrid posterior*

$$P(c_j|x) = f(c_j|x)^{\alpha} h(c_j|x)^{1-\alpha}, \quad \alpha \in [0,1],$$

is Bayes-optimal under any convex loss function $\mathcal{L}$ that preserves risk convexity in $P(c_j|x)$.

Proof Sketch. Let $\mathcal{R}(P) = \mathbb{E}_{x,y}[\mathcal{L}(P(c_j|x), y)]$ denote the expected risk under $\mathcal{L}$. Since $\mathcal{L}$ is convex and both f and h minimize $\mathcal{R}$, Jensen's inequality gives:

$$\mathcal{R}(P) \le \alpha \mathcal{R}(f) + (1-\alpha) \mathcal{R}(h).$$

Equality holds when $f = h$, implying that $P(c_j|x)$ achieves the minimal Bayes risk within the convex hull of $\{f, h\}$. Hence, the hybrid posterior remains Bayes-optimal under convex $\mathcal{L}$. □

Received 15 May 2026; accepted 02 July 2026